WEST

Help Logout Interrupt

Main Menu | Search Form | Posting Counts | Show S Numbers | Edit S Numbers | Preferences | Cases

Search Results -

Terms	Documents
L8 and probe	24

Database:
- US Patents Full-Text Database
- US Pre-Grant Publication Full-Text Database
- JPO Abstracts Database
- EPO Abstracts Database
- Derwent World Patents Index
- IBM Technical Disclosure Bulletins

Search: L9

Refine Search

Recall Text Clear

Search History

DATE: Tuesday, June 03, 2003 Printable Copy Create Case

Set Name side by side	**Query**	**Hit Count**	**Set Name** result set
DB=USPT; PLUR=YES; OP=OR			
L9	L8 and probe	24	L9
L8	l2 same on-chip	52	L8
L7	L6 and scan	4	L7
L6	L5 and probe	8	L6
L5	l2 near3 embedded	24	L5
DB=USPT,PGPB; PLUR=YES; OP=OR			
L4	(5144525| 5504756| 5574733| 5757818| 5796282| 5872795| 5968181| 5993055| 6016563| 6097232| 6182247| 6212652| 6286114)![pn]	13	L4
DB=USPT; PLUR=YES; OP=OR			
L3	L2 and l1	4	L3
L2	logic adj1 analyzer	969	L2
L1	(ton near1 david)[xa,xp]	307	L1

WEST

□ Generate Collection Print

L9: Entry 14 of 24 File: USPT Aug 22, 2000

DOCUMENT-IDENTIFIER: US 6107821 A
TITLE: On-chip logic analysis and method for using the same

Brief Summary Text (13):
The logic resources of a conventional PLD can be configured to provide a logic analysis circuit, i.e., a circuit that provides stimulus for, observes, and/or analyzes logic values in a circuit under test. However, a conventional PLD is generally heavily utilized, leaving few, if any, available logic resources free for the logic analysis circuit. This resource limitation dictates that the logic analysis circuit be external to the PLD (off-chip). However, the state data from the test nodes in the PLD must still be distributed to the external logic analysis circuit at appropriate times. In a conventional PLD, a portion of the logic resources are configured as "probe circuits", which provide both triggering functions and the transference of state data from the test nodes to various pins. Although the probe circuits consume a portion of the available logic and routing resources, this portion is much smaller than would be required by a logic analysis circuit.

Brief Summary Text (14):
FIG. 2 shows a conventional FPGA 200 having a portion of its logic and routing resources configured as probe functions. FPGA 200 is similar to FPGA 100, comprising CLBs 102a-102i surrounded by IOBs 104a-104l, a CIS 110, a configuration port 120, and a configuration control circuit 130. CLBs 102b and 102f and IOBs 104b and 104e (shown shaded) are configured to provide probe circuits for testing of the UC in the remaining CLBs and IOBs.

Brief Summary Text (15):
The use of probe circuits to gather the data for verification of a UC provides great flexibility in the selection of trigger logic and test nodes. The logic and routing resources used to provide the probe circuits are from the same pool of resources used by the active UC. Therefore, the probe circuits have direct access to the test nodes of the active UC.

Brief Summary Text (16):
However, the use of otherwise general-purpose CLBs and IOBs to create the probe circuits reduces the logic and routing resource available for the desired UC. Therefore, in heavily utilized PLDs, resource limitations may curtail the effectiveness of probe circuit testing. Either a reduced number of probe circuits may be used to perform partial testing of the UC,

Brief Summary Text (17):
or the UC itself may be only partially configured (i.e., portions of the desired logic may be eliminated from the UC), to make additional logic and routing resources available for the probe functions. Neither option is completely satisfactory. Partial testing can be time-consuming if multiple test runs must be made to cover the full range of UC operation. In addition, partial testing may fail to detect problems associated with the full UC. On the other hand, partially configuring the UC can potentially alter the performance of the UC. Further, the inclusion of probe functions in the PLD along with the UC can affect the performance of the UC. The additional gates, gate activity, and routing modifications can generate noise and signal delays in the UC, leading to erroneous test results.

Brief Summary Text (18):
Another problem associated with probe function testing is that a large number of pins may be required to transmit the state data to the external logic analyzer. Typically, one pin is used to transfer the state data from each test node being examined. The active UC requires a certain number of pins for its own data input/output activity.

Because the total number of pins in a PLD package is limited, typically either the UC must be modified or the test abridged.

Brief Summary Text (20):
An alternative method for testing the operation of a UC is the Readback process, developed by Xilinx, Inc. and used in their XC4000.TM. series FPGAs. The Readback process addresses some of the resource limitation and signal delay issues of probe circuit testing. FIG. 3 shows a simplified circuit diagram of an FPGA 300, which is consistent with the XC4000-series FPGAs from Xilinx, Inc. FPGA 300 is similar to FPGA 100 (FIG. 1), and comprises an array of CLBs 102 surrounded by a ring of IOBs 104 and interconnected by a CIS 110 including multiple PSMs 106. As in FPGA 100, CLBs 102, IOBs 104, and PSMs 106 in FPGA 300 are configured by a configuration control circuit 130 using data received through a configuration port 120. However, FPGA 300 further includes readback logic resources 302 and readback routing resources 304. Readback logic resources 302 comprise a trigger net (not shown) that can be connected to any IOB 104 by readback routing resources 304. When a low-to-high transition takes place on the trigger net, readback logic resources 302 begin shifting out a data stream that reports the configuration bits of FPGA 300. Readback logic resources 302 can be configured to also include the contents of all flip-flops and latches in FPGA 300 in the readback data stream. This data stream is fed to an IOB that routes the data stream to an external logic analyzer (not shown).

Brief Summary Text (30):
Numerous benefits accrue from this on-chip logic analysis capability. Because the multiple-context PLD can perform its own logic analysis, an "off-chip" logic analyzer is not required. Testing speed is also improved, because the data signals do not have to travel along long interconnect lines and through pins to an external logic analyzer. Testing validity is enhanced because the full UC is used, without potentially performance-affecting modifications. In addition, the resource limitation issues of conventional logic analysis methods (i.e., pin count, unused logic resources) are avoided because all the logic resources are available for the logic analysis function.

Drawing Description Text (5):
FIG. 2 is a simplified circuit diagram of a conventional FPGA configured to include probe function capability.

WEST

□ Generate Collection Print

L9: Entry 6 of 24 File: USPT Oct 1, 2002

DOCUMENT-IDENTIFIER: US 6460148 B2
TITLE: Enhanced embedded logic analyzer

Brief Summary Text (8):
One approach to debugging a hardware device within a working system is to use a separate piece of hardware equipment called a logic analyzer to analyze signals present on the pins of a hardware device. (For example, the HP1670A Series Logic Analyzer from Hewlett-Packard Company.) Typically, a number of probe wires are connected manually from the logic analyzer to pins of interest on the hardware device in order to monitor signals on those pins. The logic analyzer captures and stores these signals. However, the use of an external logic analyzer to monitor pins of a hardware device has certain limitations when it comes to debugging such a device. For example, such an external logic analyzer can only connect to and monitor the external pins of the hardware device. Thus, there is no way to connect to and monitor signals that are internal to the hardware device. Unfortunately, when programming a hardware device such as a PLD, it would be useful to be able to monitor some of these internal signals in order to debug the PLD.

Brief Summary Text (11):
In some cases, it is possible for an engineer to employ a conventional logic analyzer to study an internal signal of a PLD. This may be accomplished by, for example, an engineer modifying his design so that a normally internal signal is routed temporarily to an output pin of the PLD. The design is then recompiled. The engineer then attaches a probe to this output pin in order to monitor the "internal" signal. Unfortunately, the engineer must recompile his design and reprogram the PLD in order to view this internal signal. Also, when debugging is complete, the engineer must again rewrite the design to remove the internal signal from the output pin, recompile the design and fully reprogram the PLD again. This can be a tedious process.

Brief Summary Text (12):
Even if an engineer is successful in routing an internal signal to an output pin of a PLD, with certain integrated circuit packages it may be extremely difficult to attach an external logic analyzer. For an integrated circuit in a dual in-line package it may be relatively straightforward to attach the probes of a logic analyzer to the top of the package as long as the package is in an easily accessible location on a circuit board. However, if the package is in a difficult to reach location because of device crowding, it may be difficult to physically attach logic analyzer probes to particular output pins of interest. Even more troublesome are integrated circuits with rows of miniature contacts located on the top of the package (e.g., "flip chips"). It is difficult to attach logic analyzer probes to particular outputs of interest with this type of package. Some integrated circuit are encased in a ball grid array package with the contacts located on the bottom of the package up against the circuit board; for these packages, it may be nearly impossible to attach logic analyzer probes to these small contacts located on the underside of the package. Thus, use of an external logic analyzer has shortcomings even if an internal signal can be routed to a pin of a device.

Brief Summary Text (16):
Various prior art efforts present partial solutions, but each have their drawbacks. For example, external logic analyzers available from the Hewlett-Packard Company allow capture of signal data before a trigger condition (or breakpoint) occurs. Unfortunately, these external logic analyzers suffer from many of the disadvantages associated with external logic analyzers discussed above. Actel Corporation of Sunnyvale, Calif. provides two probes within a programmable logic device that are able to monitor two different signals, but these signals must be prespecified by the user and may not be flexibly reassigned to other signals. In addition, the Actel probes

provide constant monitoring of particular signals, but do not allow capture of relevant signal data in relation to a specified breakpoint.

Brief Summary Text (27):
The present invention provides both an apparatus and a technique by which a logic analyzer circuit is automatically embedded within a PLD, by which it captures and stores logic signals both before and after a breakpoint, and by which it unloads these signals through an interface to a computer. In a preferred embodiment, analysis of the signals is performed on the computer, with the "on-chip" logic analyzer circuit serving only to acquire the signals. The invention works especially well with a PLD because, by its very nature, a PLD is able to be programmed with a design, the design may be changed, and the PLD programmed again and again. Thus, the logic analyzer circuit may be embedded in test designs or iterations in the process of designing a final PLD. Upon successful debugging of the PLD design, the PLD chip can be reprogrammed without the logic analyzer circuit, or the circuit can be left on the chip.

Brief Summary Text (34):
In one embodiment of the present invention, a number of pins on the PLD are dedicated interface pins for communication with the user computer. Because these pins are dedicated for the interface, and are known ahead of time, they may be routed to an easily accessible location or port on a circuit board, such that a debugging interface cable may be connected from the user computer to these pins extremely easily. This technique is especially advantageous where pins or contacts of a particular integrated circuit in a package may be difficult or nearly impossible to reach. Because the embedded logic analyzer of the present invention may be configured to monitor any internal or external signals of the PLD, all of these monitored signals are available for analysis through these interface pins. In other words, it is not necessary to physically connect a probe to a particular external pin of interest because any signal within the PLD can be monitored, stored within the memory of the embedded logic analyzer and then later uploaded to the user computer for analysis through these dedicated interface pins.

Other Reference Publication (1):
Robert R. Collins, "Overview of Pentium Probe Mode, " (www.x86.org/articles/probemd/ProbeMode.htm), Aug. 21, 1998, 3 pages.

WEST

□ Generate Collection Print

L3: Entry 1 of 4 File: USPT May 13, 2003

DOCUMENT-IDENTIFIER: US 6564347 B1
TITLE: Method and apparatus for testing an integrated circuit using an on-chip logic analyzer unit

Abstract Text (1):
An on-chip logic analyzer unit. An integrated circuit includes a programmable logic analyzer unit embedded within the integrated circuit to test a function of the integrated circuit.

Primary Examiner (1):
Ton; David

Brief Summary Text (3):
An embodiment of the present invention relates to the field of testing integrated circuits and more specifically, to an on-chip logic analyzer unit (LAU).

Brief Summary Text (6):
For post-fabrication validation testing, for example, a logic analyzer may be used to perform a particular level of functional testing. In general terms, a logic analyzer is connected to a device to be tested using one or more probes. A processor in the logic analyzer executes instructions in response to commands received via a logic analyzer user interface. The executed instructions cause the logic analyzer to stimulate the device being tested via the one or more probes with one or more test signals, referred to herein as signal vectors.

Brief Summary Text (7):
In response to receiving a signal vector from the logic analyzer, the device under test generates one or more response signal vectors that are sent to the logic analyzer via the probes and may be displayed on a logic analyzer display. Pass/fail functionality of the device under test may be determined by comparing reference signal vectors stored by the logic analyzer with the response signal vectors received from the device under test.

Brief Summary Text (8):
In order to test an integrated circuit effectively, a logic analyzer should be capable of applying and receiving signal vectors at a rate at least equal to the desired operating speed of the integrated circuit being tested. Thus, as the operating clock frequency of each generation of integrated circuits continues to increase, logic analyzers used to test integrated circuits must also become faster.

Brief Summary Text (9):
As logic analyzers (and other similar testers) increase in speed, however, they also increase in cost and design complexity. Additionally, most integrated circuit manufacturers purchase logic analyzers from external vendors who may have difficulty producing products that meet the integrated circuit manufacturers technical requirements in a timely manner. Further, using logic analyzer probes, it may not be possible to access all of the signals that are desirable to test.

Brief Summary Text (13):
Built-In Self-Test (BIST) capability (another DFT feature) is also included in many integrated circuit devices. BIST is typically implemented as a microcoded program in microcode Read Only Memory (ROM) to exercise the microarchitectural elements of the host integrated circuit to determine whether they logically operate as specified. Because BIST is a firmware tool, it is relatively inflexible. Further, BIST typically only produces a pass/fail indication such that further and more extensive testing (possibly using an external logic analyzer) must be performed to determine a

contributing circuit failure, for example.

Brief Summary Text (15):
A method and apparatus for testing an integrated circuit are described. In accordance with one embodiment, an apparatus comprises a programmable logic analyzer unit (LAU) embedded within an integrated circuit. The programmable LAU tests a function of the integrated circuit.

Detailed Description Text (3):
For one embodiment, an integrated circuit may be tested using an on-chip logic analyzer unit (LAU). In this manner, for validation testing, for example, an external logic analyzer does not need to be used.

Detailed Description Text (4):
The on-chip LAU may provide most of the functionality of current, external logic analyzers while being capable of scaling with the speed of the integrated circuit being tested. Further, the on-chip LAU of one embodiment provides for more comprehensive on-chip testing capabilities than prior embedded design for testability (DFT) features. Additionally, by being programmable, test routines can be added after manufacture of the integrated circuit to be tested.

Detailed Description Text (17):
Using some or all of the above-described circuit blocks and components, the on-chip LAU 105 of one embodiment can perform many of the same functions as currently available logic analyzers that are external to a chip to be tested. Some examples of these capabilities are described in more detail below with continuing reference to FIGS. 1 and 2.

Detailed Description Text (36):
As described above, the on-chip LAU provides many of the capabilities of currently available, external logic analyzers. The on-chip LAU of one embodiment, however, scales in performance with host integrated circuit technology. More specifically, the on-chip LAU is fabricated using the same technology as the host chip, may be clocked using host chip clocks and, thus, can keep pace with the technology to be tested.

Detailed Description Text (37):
Further, the functions provided by external logic analyzers change slowly as compared to corresponding advances in integrated circuit functionality and technology. Assuming this trend continues, the number of transistors used to implement an on-chip LAU in accordance with various embodiments will not increase as quickly as the number of transistors in host integrated circuits to be tested. Thus, the on-chip LAU may take up a smaller percentage of chip real estate as host chips continue to increase in complexity. The LAU of one embodiment may be implemented as a standard cell that can easily be used in the design of many integrated circuits.

Detailed Description Text (39):
Using embedded LAUs in integrated circuits in accordance with the invention, integrated circuit manufacturers and testers may be able to realize significant savings related to test equipment by obviating a need for costly, external logic analyzers. As a need for external logic analyzers is reduced, so are the costs associated with maintenance, repair, etc.

CLAIMS:

1. An integrated circuit comprising: a programmable logic analyzer unit embedded within the integrated circuit, the programmable logic analyzer unit to test a function of the integrated circuit; and programmable clock delay logic being programmable by the logic analyzer unit to generate a shifted clock signal in response to receiving an input clock signal.

2. The integrated circuit of claim 1 further comprising: a memory coupled to the programmable logic analyzer unit, the memory to store a vector for use by the programmable logic analyzer unit during testing of the integrated circuit.

3. The integrated circuit of claim 1 further comprising: input/output circuitry coupled to the programmable logic analyzer unit, an input of the integrated circuit and an output of the integrated circuit, the programmable logic analyzer to stimulate the input and receive a response from the output via the input/output circuitry.

5. The integrated circuit of claim 1 wherein the programmable logic analyzer unit comprises: a logic analyzer control unit; and a vector storage memory coupled to the logic analyzer control unit.

6. The integrated circuit of claim 5 wherein the logic analyzer control unit comprises: a pattern generator unit to generate a test pattern for the integrated circuit; and a programmable state machine coupled to the pattern generator unit to control a test sequence during testing of the integrated circuit.

8. The integrated circuit of claim 5 further comprising: a host memory coupled to the programmable logic analyzer unit, wherein the vector storage memory is included within the host memory.

10. The integrated circuit of claim 1 further comprising: an internal scan chain coupled to the programmable logic analyzer unit, the internal scan chain to interconnect the programmable logic analyzer unit and a circuit to be tested on the integrated circuit.

11. The integrated circuit of claim 1 wherein the programmable logic analyzer unit is coupled to an internal circuit on the integrated circuit, the internal circuit being inaccessible by a pin on the integrated circuit, the programmable logic analyzer unit being capable of testing the functionality of the internal circuit.

12. The integrated circuit of claim 1 wherein the programmable logic analyzer unit comprises a signal typing control unit, the signal typing control unit to enable the logic analyzer unit to drive and sense a plurality of different types of signals, the plurality of different types of signals including at least one of edge-triggered, level-sensitive, positive precharged, negative precharged and tri-stated signals.

13. A method comprising: selectively coupling a logic analyzer control unit on an integrated circuit to a first circuit on the integrated circuit to be tested; the logic analyzer unit retrieving a signal vector from a memory on the integrated circuit; the logic analyzer unit applying the signal vector to an input of the first circuit on the integrated circuit; programmably shifting an edge of a clock signal using circuitry on the integrated circuit to generated a shifted clock signal, the shifted clock signal being used to time an aspect of testing the first circuit; and the logic analyzer unit sensing a response vector from an output of the first circuit at a second circuit on the integrated circuit.

20. A desktop computer system comprising: a bus; an input device coupled to the bus; a display device coupled to the bus; and a processor coupled to the bus, the processor comprising: a host memory; a programmable logic analyzer unit coupled to the host memory, the programmable logic analyzer unit to test a function of the processor in response to input via the input device; and clock delay logic programmable by the logic analyzer unit to provide a shifted clock signal in response to receiving an input clock signal, the shifted clock signal to be provided to the programmable logic analyzer unit.

21. The computer system of claim 20 wherein the logic analyzer is programmable via the input device, and wherein a result of testing the function of the processor with the programmable logic analyzer is displayed using the output device.

22. The computer system of claim 21 wherein the host memory is further to receive and store after manufacture of the processor a program to be used by the programmable logic analyzer unit to test the function of the processor.

23. The computer system of claim 20 wherein the shifted clock signal is used by the programmable logic analyzer unit to test a margin associated with the function of the processor.

24. The computer system of claim 20 wherein the programmable logic analyzer unit further comprises a signal typing control unit, the signal typing control unit to be used to enable the programmable logic analyzer unit to drive and sense a plurality of different types of signals, the plurality of different types of signals including at least one of edge-triggered, level-sensitive, positive precharged, negative precharged and tri-stated signals.

WEST

□ Generate Collection Print

L5: Entry 22 of 24 File: USPT Jan 18, 2000

DOCUMENT-IDENTIFIER: US 6016563 A
TITLE: Method and apparatus for testing a logic design of a programmable logic device

Abstract Text (1):

An apparatus and method are provided for the development, testing and verification of a logic design of a programmable logic device in a real-time user environment to simplify the development of the programmable logic device and associated systems. The apparatus comprises an emulation programmable logic device based on the same family and package of the target programmable logic device. The adapter further comprises a plurality of individually programmable switches for selectively coupling the emulation device to the target device or to a logic device substituting for the target device. The apparatus further comprises a controller, which configures the switches based on control signals received from a host computer system, such that a stimulus applied to the input pins of the target or substitute device are also applied concurrently to the corresponding input pins of the emulation device. The switches are further configured so that the output pins of the emulation device are unloaded from all of their normal functions and are output only as test points. Signal values at internal nodes of the emulation device in response to the stimulus can be traced out via the test points without disturbing the target or substitute device. The apparatus permits dynamic and independent programming and reprogramming of both the target and mirror devices, such that different internal nodes can be traced out quickly and easily. The apparatus may further include an embedded programmable stimulus generator and an embedded logic analyzer.

Drawing Description Text (9):

FIG. 7 illustrates an embodiment of an emulation system which includes an embedded logic analyzer module.

Detailed Description Text (22):

An emulation system of the present invention may also include an embedded logic analyzer. As an example, FIG. 7 shows an embodiment similar to that of FIG. 6 but also including an embedded logic analyzer module 66. The embedded logic analyzer module 66 connects to test points 20A and 20B in parallel with the programmable pattern generator 64. The controller 12 appropriately configures the logic analyzer 66 using signal 69, based on signals received from the host computer 30.

WEST

End of Result Set

□ Generate Collection Print

L5: Entry 24 of 24 File: USPT Apr 7, 1998

DOCUMENT-IDENTIFIER: US 5737520 A
TITLE: Method and apparatus for correlating logic analyzer state capture data with associated application data structures

Brief Summary Text (20):
The methods of the present invention are operable on a general purpose computer. Such a computer may be either embedded within a logic analyzer or external to a logic analyzer operating in a post processing mode to analyze and display buffer and protocol information in the captured states retrieved from a logic analyzer. In the alternative, portions of the present invention may be embedded within custom hardware assist electronic circuits to enhance the performance of the analysis and recognition methods of the present invention. Such custom assist circuits may be integrated within the user interface control circuits and software of a logic analyzer.

Drawing Description Text (3):
FIG. 2 is a block diagram depicting a self-contained logic analyzer including an embedded processor and display on which the methods of the present invention are operable;

Detailed Description Text (11):
One of ordinary skill in the art will recognize that the methods of the present invention may be applied within general purpose logic analyzers as well as embedded within special purpose protocol or bus analyzer devices or software modules.

WEST

□ Generate Collection Print

L6: Entry 1 of 8 File: USPT May 13, 2003

DOCUMENT-IDENTIFIER: US 6564347 B1
TITLE: Method and apparatus for testing an integrated circuit using an on-chip logic analyzer unit

Abstract Text (1):
An on-chip logic analyzer unit. An integrated circuit includes a programmable logic analyzer unit embedded within the integrated circuit to test a function of the integrated circuit.

Brief Summary Text (6):
For post-fabrication validation testing, for example, a logic analyzer may be used to perform a particular level of functional testing. In general terms, a logic analyzer is connected to a device to be tested using one or more probes. A processor in the logic analyzer executes instructions in response to commands received via a logic analyzer user interface. The executed instructions cause the logic analyzer to stimulate the device being tested via the one or more probes with one or more test signals, referred to herein as signal vectors.

Brief Summary Text (7):
In response to receiving a signal vector from the logic analyzer, the device under test generates one or more response signal vectors that are sent to the logic analyzer via the probes and may be displayed on a logic analyzer display. Pass/fail functionality of the device under test may be determined by comparing reference signal vectors stored by the logic analyzer with the response signal vectors received from the device under test.

Brief Summary Text (9):
As logic analyzers (and other similar testers) increase in speed, however, they also increase in cost and design complexity. Additionally, most integrated circuit manufacturers purchase logic analyzers from external vendors who may have difficulty producing products that meet the integrated circuit manufacturers technical requirements in a timely manner. Further, using logic analyzer probes, it may not be possible to access all of the signals that are desirable to test.

Brief Summary Text (15):
A method and apparatus for testing an integrated circuit are described. In accordance with one embodiment, an apparatus comprises a programmable logic analyzer unit (LAU) embedded within an integrated circuit. The programmable LAU tests a function of the integrated circuit.

Detailed Description Text (19):
A designer of the host integrated circuit chip 100, for example, may choose the types and locations of connections between the LAU 105 and internal host chip 100 circuitry based on circuits and signal lines desired to be tested. In this manner, the LAU 105 of some embodiments is capable of testing circuit implementations, signal lines etc. that may not easily be tested by other types of test circuitry. Thus, in contrast to BIST firmware, for example, the LAU 105 may probe arbitrary wires in the circuit implementation of a microarchitecture rather than being limited to testing the logical operation of microarchitectural elements. Further, the LAU 105 of some embodiments is capable of examining physical as well as logical aspects of the events in wires and memory elements of the host IC 100. Additionally, the programmable nature of the LAU 105 makes it quite flexible in terms of varying tests to be performed, etc. in comparison to BIST routines. While BIST routines produce a pass/fail indication, the LAU 105 may be used as a debugging tool to observe operation of the host IC 100 at a lower level.

CLAIMS:

1. An integrated circuit comprising: a programmable logic analyzer unit embedded within the integrated circuit, the programmable logic analyzer unit to test a function of the integrated circuit; and programmable clock delay logic being programmable by the logic analyzer unit to generate a shifted clock signal in response to receiving an input clock signal.

WEST

□ Generate Collection Print

L6: Entry 2 of 8 File: USPT Oct 1, 2002

DOCUMENT-IDENTIFIER: US 6460148 B2
TITLE: Enhanced embedded logic analyzer

Abstract Text (1):

Embedding a logic analyzer in a programmable logic device allows signals to be captured both before and after a trigger condition (breakpoint). A logic analyzer embedded within a PLD captures and stores logic signals. It unloads these signals for viewing on a computer. Using an electronic design automation (EDA) software tool running on a computer system, an engineer specifies signals of the PLD to be monitored, a breakpoint, total number of samples to be stored, number of samples to be captured after the breakpoint occurs, and a system clock signal. The EDA tool automatically inserts the logic analyzer into the electronic design of the PLD which is compiled and downloaded to configure the PLD. Using an interface connected between the PLD and the computer, the EDA tool commands the embedded logic analyzer to run. Signals are stored continuously while running in a ring buffer RAM memory. Once the breakpoint occurs, more samples are captured if desired, in addition to those signals captured before breakpoint. The EDA tool directs the logic analyzer to unload the data from its capture buffer for display on a computer. The breakpoint and sample number can be changed without recompiling. A JTAG port controls the logic analyzer. Inputs and outputs of the logic analyzer are routed to unbonded JTAG-enabled I/O cells. Alternatively, a user-implemented test data register provides a JTAG-like chain of logic elements through which control and output information is shifted. Stimulus cells provide control information to the logic analyzer, and sense cells retrieve data from the logic analyzer.

Parent Case Text (1):

This application claims priority of U.S. provisional patent application No. 60/065,602, filed Nov. 18, 1997, entitled "Enhanced Embedded Logic Analyzer" which is incorporated by reference. This application is a continuation of U.S. patent application Ser. No. 09/186,607 filed Nov. 16, 1998 now U.S. Pat. No. 6,286,114, which in turn is a continuation-in-part of U.S. patent application Ser. No. 08/958,435, filed Oct. 27, 1997 now U.S. Pat. No. 6,182,247, entitled "Embedded Logic Analyzer For A Programmable Logic Device" which is incorporated by reference.

Parent Case Text (2):

This application is related to U.S. Pat. No. 6,247,147, entitled "Enhanced Embedded Logic Analyzer," which is hereby incorporated by reference.

Brief Summary Text (2):

The present invention relates generally to analysis of a hardware device in connection with a computer system. More specifically, the present invention relates to a logic analyzer that is automatically embedded within a hardware device for purposes of debugging.

Brief Summary Text (8):

One approach to debugging a hardware device within a working system is to use a separate piece of hardware equipment called a logic analyzer to analyze signals present on the pins of a hardware device. (For example, the HP1670A Series Logic Analyzer from Hewlett-Packard Company.) Typically, a number of probe wires are connected manually from the logic analyzer to pins of interest on the hardware device in order to monitor signals on those pins. The logic analyzer captures and stores these signals. However, the use of an external logic analyzer to monitor pins of a hardware device has certain limitations when it comes to debugging such a device. For example, such an external logic analyzer can only connect to and monitor the external pins of the hardware device. Thus, there is no way to connect to and monitor signals that are internal to the hardware device. Unfortunately, when programming a hardware device such as a PLD, it would be useful to be able to monitor some of these internal signals in order to debug the PLD.

Brief Summary Text (11):
In some cases, it is possible for an engineer to employ a conventional logic analyzer to study an internal signal of a PLD. This may be accomplished by, for example, an engineer modifying his design so that a normally internal signal is routed temporarily to an output pin of the PLD. The design is then recompiled. The engineer then attaches a probe to this output pin in order to monitor the "internal" signal. Unfortunately, the engineer must recompile his design and reprogram the PLD in order to view this internal signal. Also, when debugging is complete, the engineer must again rewrite the design to remove the internal signal from the output pin, recompile the design and fully reprogram the PLD again. This can be a tedious process.

Brief Summary Text (12):
Even if an engineer is successful in routing an internal signal to an output pin of a PLD, with certain integrated circuit packages it may be extremely difficult to attach an external logic analyzer. For an integrated circuit in a dual in-line package it may be relatively straightforward to attach the probes of a logic analyzer to the top of the package as long as the package is in an easily accessible location on a circuit board. However, if the package is in a difficult to reach location because of device crowding, it may be difficult to physically attach logic analyzer probes to particular output pins of interest. Even more troublesome are integrated circuits with rows of miniature contacts located on the top of the package (e.g., "flip chips"). It is difficult to attach logic analyzer probes to particular outputs of interest with this type of package. Some integrated circuit are encased in a ball grid array package with the contacts located on the bottom of the package up against the circuit board; for these packages, it may be nearly impossible to attach logic analyzer probes to these small contacts located on the underside of the package. Thus, use of an external logic analyzer has shortcomings even if an internal signal can be routed to a pin of a device.

Brief Summary Text (13):
U.S. patent application Ser. No. 08/958,435 entitled "Embedded Logic Analyzer For A Programmable Logic Device" discloses an advantageous apparatus and techniques that allow an embedded logic analyzer to flexibly analyze internal signals of interest in an electronic design, such as within a programmable logic device (PLD). Nevertheless, there is room for improvement in the analysis of internal signals of a PLD for debugging purposes.

Brief Summary Text (16):
Various prior art efforts present partial solutions, but each have their drawbacks. For example, external logic analyzers available from the Hewlett-Packard Company allow capture of signal data before a trigger condition (or breakpoint) occurs. Unfortunately, these external logic analyzers suffer from many of the disadvantages associated with external logic analyzers discussed above. Actel Corporation of Sunnyvale, Calif. provides two probes within a programmable logic device that are able to monitor two different signals, but these signals must be prespecified by the user and may not be flexibly reassigned to other signals. In addition, the Actel probes provide constant monitoring of particular signals, but do not allow capture of relevant signal data in relation to a specified breakpoint.

Brief Summary Text (17):
Therefore it would be desirable to have an apparatus and technique that would allow a logic analyzer embedded within a programmable logic device to flexibly capture internal signals both before and after a specified breakpoint.

Brief Summary Text (18):
Furthermore, it would be desirable to have an apparatus and technique that would efficiently and flexibly control a logic analyzer embedded within a programmable logic device. As explained below, although various options are available for controlling such an embedded logic analyzer, none of the prior art techniques are optimal. By way of background, a brief explanation of the design and manufacturing phases of a PLD and a circuit board will be provided first.

Brief Summary Text (19):
As described earlier in this section, a design engineer designs a PLD and programs such a device using an electronic design automation tool. In the course of this design phase, the design engineer may perform numerous design-program-debug iterations before

the design is complete and the PLD ready for mass manufacturing. The design engineer often uses a simulation and/or a timing analysis to assist in debugging the electronic design of the PLD. It is also conceivable that a design engineer would use an embedded logic analyzer (such as disclosed in U.S. patent application Ser. No. 08/958,435) to troubleshoot the design. Once the design of the PLD is complete to the design engineer's satisfaction, the design is handed off to a product engineer for the manufacturing phase.

Brief Summary Text (21):
A full board test may then be performed to test the traces, solder connections, and other physical interfaces between components on the board. It should be pointed out that a board test may also be performed before any devices on the board are programmed or configured. It is common to use a JTAG port of a PLD or other device to test the traces and solder connections of a board during this board test. Once physical connections are tested, a complete functional test of the board is then formed to test the overall functionality of the board (i.e., to ensure that particular inputs produce the outputs expected). At this point, if a failure is detected it may be necessary to debug a particular PLD while on the board. For failures more difficult to track down, it may even be necessary to remove a PLD from the board to be debugged. In these circumstances, as previously explained, it is desirable to have an embedded logic analyzer within the PLD to facilitate debugging. During any debugging of the PLD using an embedded logic analyzer, it is necessary in some fashion to control the embedded logic analyzer, i.e., to provide it with commands and data and to receive captured data and status from it. Although various options are available, none are currently optimal.

Brief Summary Text (22):
For example, it may be possible to use existing input/output pins of a device to provide a control interface. Unfortunately, a particular design may not have enough extra input/output pins available through which an interface can be provided to control an embedded logic analyzer. It can be undesirable to require that a customer purchasing a PLD not use a certain number of input/output pins simply because the PLD may not have been designed correctly and might have to be debugged at some point.

Brief Summary Text (24):
Therefore, an apparatus and technique are further desirable that would provide simple, efficient and flexible control of an embedded logic analyzer. It would further be desirable for such a control apparatus and technique to allow testing of a PLD on a circuit board in a real-world environment.

Brief Summary Text (26):
To achieve the foregoing, and in accordance with the purpose of the present invention, a technique for embedding a logic analyzer in a programmable logic device is disclosed that allows capture of specified signal data both before and after a specified breakpoint. Also disclosed are techniques for controlling an embedded logic analyzer using a JTAG port.

Brief Summary Text (27):
The present invention provides both an apparatus and a technique by which a logic analyzer circuit is automatically embedded within a PLD, by which it captures and stores logic signals both before and after a breakpoint, and by which it unloads these signals through an interface to a computer. In a preferred embodiment, analysis of the signals is performed on the computer, with the "on-chip" logic analyzer circuit serving only to acquire the signals. The invention works especially well with a PLD because, by its very nature, a PLD is able to be programmed with a design, the design may be changed, and the PLD programmed again and again. Thus, the logic analyzer circuit may be embedded in test designs or iterations in the process of designing a final PLD. Upon successful debugging of the PLD design, the PLD chip can be reprogrammed without the logic analyzer circuit, or the circuit can be left on the chip.

Brief Summary Text (28):
In one embodiment of the invention, using an electronic design automation (EDA) software tool running on a computer system, an engineer specifies signals of the PLD to be monitored, specifies the number of samples to be captured, specifies a system clock signal, and specifies not only a breakpoint, but also the number of samples needed prior to the breakpoint. (Alternatively, total samples could be specified and/or samples needed after a breakpoint.) The EDA tool then automatically inserts the logic analyzer circuit into the electronic design of the PLD which is compiled and downloaded to configure the PLD. Using an interface connected between the PLD and the computer,

the EDA tool communicates with the embedded logic analyzer in order to instruct the logic analyzer to run and to begin capturing data. Once a breakpoint occurs, the logic analyzer determines if more samples need to be captured after the breakpoint. The EDA tool then directs the logic analyzer to unload the data from sample memory and then displays the data on the computer. The logic analyzer circuit may then run again to capture another sequence of sample values.

Brief Summary Text (30):
Often, a JTAG port is used either to program a PLD or to assist with testing a circuit board on which PLDs are located. Advantageously, it is realized that a JTAG port has traditionally been unused during the design and debugging of a particular PLD. Thus, it is further realized that a JTAG port on a PLD is under utilized and may be used during debugging of a PLD as a means of communicating with and controlling an embedded logic analyzer of the present invention. Advantageously, the standard JTAG port is used to facilitate debugging of a programmable logic device that includes an embedded logic analyzer. Use of a JTAG port avoids adding dedicated debugging control pins. In a first embodiment for controlling an embedded logic analyzer using a JTAG port, inputs and outputs of the logic analyzer are routed to unbonded JTAG-enabled I/O cells. Cells that will provide control signals are tricked into thinking they are in INTEST mode so that signals may be input, yet the rest of the device operates as in a real-world environment. In a second embodiment, a user-implemented test data register provides a JTAG-like chain of logic elements through which control and output information is shifted. Stimulus cells provide control information to the logic analyzer, and sense cells retrieve data from the logic analyzer.

Brief Summary Text (31):
The present invention provides many advantages over the prior art. Use of an embedded logic analyzer in a PLD allows debugging of the device in the system in which it is operating and under the actual conditions that might produce a malfunction of the PLD. The technique of the present invention automatically embeds a logic analyzer circuit into a PLD so that an engineer may debug any logic function within the device. The embedded logic analyzer is able to capture any internal signals specified by the engineer; the breakpoint can also include any specified internal signals. Through the use of memory within the embedded logic analyzer and an interface to the computer, any number and depth of signals can be monitored within the device and then transmitted to the computer at a later time for analysis. In one embodiment of the invention, a JTAG port is used to program the embedded logic analyzer and to transmit captured signal information to the computer.

Brief Summary Text (32):
Advantageously, while debugging a PLD design in a system, an engineer may use the EDA tool to specify new signals to monitor and/or new breakpoints. The engineer can then reprogram the device while it is within its intended system with a modified logic analyzer circuit very rapidly in order to debug a different portion of the device or to change the triggering conditions. This ability to reprogram an embedded logic analyzer on the fly has many advantages over built-in debugging hardware on custom chips that may not be dynamically reprogrammed. This ability to reprogram also has advantages over external logic analyzers that can only monitor the external pins of a hardware device. Furthermore, once an engineer has finished debugging the device with the embedded logic analyzer, the EDA tool may be used to generate a final configuration output file without the logic analyzer that represents the engineer's final working design. Thus, the logic analyzer need not be part of the final design and take up space on the PLD.

Brief Summary Text (33):
The present invention is applicable to a wide range of hardware devices, and especially to PLDs. A PLD in particular may be implemented using a wide variety of technologies, including SRAM technology and EEPROM technology. PLDs based upon SRAM technology are especially advantageous in that they may have additional embedded memory that can be used by the embedded logic analyzer to capture a large number of, and a greater depth of signals. Furthermore, an embedded logic analyzer that is designed and inserted automatically by an EDA tool means that an engineer does not require an external logic analyzer as a separate piece of equipment. Furthermore, the engineer may use the computer on which he or she is creating a design for the PLD to also control and configure the embedded logic analyzer and to review its results.

Brief Summary Text (34):
In one embodiment of the present invention, a number of pins on the PLD are dedicated interface pins for communication with the user computer. Because these pins are dedicated for the interface, and are known ahead of time, they may be routed to an

easily accessible location or port on a circuit board, such that a debugging interface cable may be connected from the user computer to these pins extremely easily. This technique is especially advantageous where pins or contacts of a particular integrated circuit in a package may be difficult or nearly impossible to reach. Because the embedded logic analyzer of the present invention may be configured to monitor any internal or external signals of the PLD, all of these monitored signals are available for analysis through these interface pins. In other words, it is not necessary to physically connect a probe to a particular external pin of interest because any signal within the PLD can be monitored, stored within the memory of the embedded logic analyzer and then later uploaded to the user computer for analysis through these dedicated interface pins.

Brief Summary Text (35):
Additionally, an embedded logic analyzer can be used with PLDs that are configured to near capacity. An engineer can temporarily remove a portion of the design unrelated to the problem under analysis, embed a logic analyzer circuit, and then debug the PLD. Once the PLD has been debugged, the engineer may then remove the embedded logic analyzer and reinsert that section of the design that he had temporarily removed.

Drawing Description Text (5):
FIGS. 3A and 3B are a flowchart describing one technique by which a logic analyzer is programmed, embedded within a device, captures data and dumps data for viewing by a user.

Drawing Description Text (7):
FIG. 5 is another view of the block diagram of FIG. 1, showing a programmable logic device having an embedded logic analyzer within an electronic system.

Drawing Description Text (8):
FIG. 6 is a more detailed block diagram of a programmable logic device having an embedded logic analyzer.

Drawing Description Text (9):
FIG. 7 illustrates an embedded logic analyzer showing its inputs and outputs according to one embodiment of the present invention.

Drawing Description Text (10):
FIG. 8 is a block diagram of an embedded logic analyzer circuit according to one embodiment of the present invention.

Drawing Description Text (11):
FIG. 9 is a symbolic view of the operation of the control state machine of the embedded logic analyzer.

Drawing Description Text (13):
FIG. 11 illustrates a first embodiment by which a JTAG port controls an embedded logic analyzer using unbonded I/O cells.

Drawing Description Text (16):
FIG. 14 illustrates a second embodiment by which a JTAG port controls an embedded logic analyzer using a test data register.

Drawing Description Text (19):
FIGS. 17A and 17B illustrate an alternative embodiment in which any number of logic analyzers embedded within a device are controlled using a JTAG port.

Detailed Description Text (22):
Within the context of the present invention, any of the above compile techniques may be modified in order to produce an embedded logic analyzer. As will be discussed in greater detail below with reference to FIG. 4, the compilation of a PLD is modified in order to insert a logic analyzer into a user's design.

Detailed Description Text (25):
Embedded Logic Analyzer

Detailed Description Text (29):
These signals to be monitored may be specified in a wide variety of ways. By way of example, a hierarchical path name for each signal may be specified, or a graphical user interface may be used to view a particular design file and to select a signal or point

from within that file to be monitored. In one alternative embodiment, the user may also specify which pins of the device will be used as an interface to the user computer, i.e., those pins to be used to send control information to the embedded logic analyzer within the PLD and to upload captured information from the logic analyzer to the user computer. Preferably, though, the pins to be used as an interface are already known, such as a JTAG port of a device.

Detailed Description Text (33):
In step 114, a breakpoint is specified. A breakpoint may include any number of signals to monitor and the logic levels that those signals must have to trigger the logic analyzer, i.e., the breakpoint describes a particular state of the device. A breakpoint may be as simple as one signal changing state, or may be a complex pattern of signals or a sequence of patterns that occur before to trigger the logic analyzer. Also, a breakpoint need not be specified in all cases; if not, the logic analyzer immediately begins capturing data upon running. Advantageously, the breakpoint can be changed at any time by the user through the use of the EDA tool, and a new breakpoint can be downloaded to the embedded logic analyzer in the device without having to recompile all of the device design files. By allowing breakpoints to be changed rapidly for a device within a system, debugging is much more efficient. Advantageously, the present invention can acquire data not only after the breakpoint, but also before it occurs.

Detailed Description Text (34):
In step 115 the user specifies the number of data samples to be captured prior to the breakpoint. Advantageously, the user may specify any number of samples to be captured prior to the breakpoint occurring, thus allowing later analysis of these prior signals to help determine the cause of a failure, error or other condition. As explained below in FIGS. 8-10, implementation of the embedded logic analyzer allows samples to be stored continuously which provides a user with any number of samples needed prior to the breakpoint. Alternatively, a user may specify the number of samples needed after the breakpoint. Because the total number of samples to be captured has been specified in step 110, it is straightforward to calculate prior samples needed based upon later samples needed, and vice-versa. The user may also specify samples needed prior to the breakpoint and samples needed after the breakpoint; total samples to be captured can then be calculated automatically.

Detailed Description Text (36):
Once the user has specified how he or she wishes the embedded logic analyzer to function, the complete design is compiled. In step 116, the user issues a compile command to compile the user's device design along with the logic analyzer design that has been specified. In a preferred embodiment of the invention, the design files are not modified during this process. The logic analyzer design is incorporated into the output files produced. In one specific embodiment, the process shown in FIG. 4 may be used to implement step 116. Of course, other similar techniques may also be used.

Detailed Description Text (37):
The result of this step is a new output file that includes the user's design with an embedded logic analyzer. A technique by which an EDA tool may insert a custom logic analyzer in a design will be discussed in greater detail below with reference to FIG. 4. Once the new output file has be en generated, in step 118 the device is reprogrammed within its system using the new output file.

Detailed Description Text (39):
The cable may attach directly to these pins, or, the signals from these pins may be routed to an easily accessible location or port on the board to which the debugging cable may easily attach. The cable will be used to transmit instructions from the computer to the embedded logic analyzer, and also to upload captured information from the logic analyzer to the computer. As discussed below, FIG. 5 shows a PLD cont aining both a user design and an embedded logic analyzer within an electronic system. A cable 28 is shown connecting the elec tronic system to an external computer.

Detailed Description Text (40):
In step 122 the user through the EDA tool requests the embedded logic analyzer to begin running with an appropriate command. Once the logic analyzer begins to run, in step 124 it begins to continuously capture data from the signals that have been specified to be monitored. Preferably, the user then manipulates the system to duplicate previous malfunctions that the user wishes to analyze The captured data is stored within memory of the PLD, and is preferably stored within dedicated memory with in the embedded logic analyzer itself. Step 126 determines weth point has occurred. In other words, the logic analyzer determines whether the state of the signals specified to be monitored are

equivalent to the e breakpoint that the user has specified. If not, then the logic analyzer continues to capture data. If so, step 128 determines whether more samples should be captured and stored after the breakpoint. Step 128 may be implemented by comparing the total number of samples desired with the number of samples that have already been stored prior to the breakpoint. If more samples are to be stored, then in step 130 the logic analyzer continues to store the desired number of samples after the breakpoint.

Detailed Description Text (46):
The actual gate level representation of a particular logic analyzer circuit will depend upon the particular device in which the logic analyzer will be embedded. By way of example, the hardware device in which to embed the logic analyzer may include any of the PLD devices available from Altera Corporation. In particular, any of the FLEX 10K, FLEX 8000, MAX 9000, or MAX 7000 devices work well. Each of these particular devices may have different features that would affect how a gate level representation for a logic analyzer is produced. For example, for a FLEX 10K device with relatively large embedded memory sections, the is embedded memory is particularly well suited for implementing a large FIFO (first in first out) memory for the logic analyzer. For a device such as the FLEX 8000 without embedded memory, the memory elements (such as SRAM flip-flops) of logic cells may be used for the memory of the logic analyzer but the FIFO buffer may have to be divided over multiple cells if the memory in a single cell is not sufficiently large to accommodate the e buffer. Similarly, a device based upon EEPROM technology may also use one or more of its logic cells for the logic analyzer's buffer. A device having large embedded memory works particularly well with the present invention because of the larger capacity for signal storage. Thus, step 206 produces a representation for a logic analyzer circuit that is to be connected to the user's design.

Detailed Description Text (49):
Of course, another embodiment of a logic analyzer circuit may use different signals and/or a greater or fewer number of interface signals. In a preferred embodiment of the invention, these interface signals to and from the logic analyzer are connected to dedicated pins on the PLD reserved for this purpose. Thus, a user will know to which pins the debugging cable should be attached. As noted, these pins not only control the embedded logic analyzer, but also receive data from it. In other embodiments, these dedicated pins may be routed to another part of the circuit board for easy attachment of a cable. In this fashion, the logic for the logic analyzer circuit created in step 206 is connected to both the user design and to interface pins of the PLD for communication with the user computer.

Detailed Description Text (50):
In step 210 the complete design created in step 208 is placed and rout ed in a fashion that will be appreciated by those of skill in the art. The output of the place and rout e step is then input to step 212 in which the output file is assembled. This output file may then be downloaded to a PLD in order to program it. Once a PLD has been programmed with this file, a user may begin use of the embedded logic analyzer in order to debug the device.

Detailed Description Text (51):
FIG. 5 is another view of programmable logic development system 10 of FIG. 1, showing a programmable logic device having an embedded logic analyzer within an electronic system. System 10 shows an electronic system 252 connected to computer system A 18 via cable 28 or other connective appliance. Electronic system 252 includes PLD 16, a component of the electronic system. PLD 16 potentially shares one or more electronic connect ions 254 with the other components and elements the at make up the electronic system. PLD 16 has been configured with a user logic design 256 and an embedded logic analyzer 260. User logic 256 is configured with a design according to the methodology described in FIG. 2, or any other suitable design methodology. Embedded logic analyzer 260 has been incorporated into PLD 16 according to one embodiment of the invention described in FIGS. 3A and 3B.

Detailed Description Text (52):
Logical connections 262 allow signals from user logic 256 to be transmitted to logic analyzer 260. These signals may include a system clock, trigger signals, signals to monitor, etc. Pins of PLD 16 are used to connect interface signals 264 from the logic analyzer to corresponding connections 266 in electronic system 252. Cable 28 is used to connect these interface signals to computer 18. Alternatively, computer 18 may be directly connected to PLD 16 to transmit interface signals 264 to the PLD. In this manner, computer 18 transmits commands and other information to embedded logic analyzer

260, and receives information from the logic analyzer without directly interrupting or affecting the functional operation of electronic system 252. PLD 16 is thus configured to perform both the functions of user logic 256 and embedded logic analyzer 260.

Detailed Description Text (53):
Those of skill in the art will appreciate that the actual interface used between logic analyzer 260 and an external computer system may take a variety of forms. The embedded logic analyzer as herein described may be controlled by an outside computer using any suitable interface on the PLD. Furthermore, the exact number of pins on PLD 16 used to control logic analyzer 260 and to receive data from it may vary depending upon the device being programmed, the overall board design, etc. It will further be appreciated that pins used may be flexibly assigned, or that dedicated interface pins may be used. For example, interface signals 264 that provide an interface between logic analyzer 260 and external pins of PLD 16 may be implemented as described in the above-referenced U.S. patent application Ser. No. 08/958,435. Other techniques for controlling an embedded logic analyzer and for receiving output data may also be used.

Detailed Description Text (54):
FIG. 6 illustrates another view of PLD 16 showing a preferred embodiment for controlling a logic analyzer using the JTAG port of the device in which the logic analyzer is embedded. Not shown for clarity within PLD 16 is user logic 256. In this preferred embodiment, interface signals 264 are implemented using a JTAG port 272 in conjunction with control logic 274 and signals 275. A JTAG (Joint Test Action Group) port 272 is implemented under the IEEE 1149.1 standard and is known to those of skill in the art. Control logic 274 provide buffering between logic analyzer 260 and JTAG port 272 for particular signals that are described below in FIG. 7. More specifically, control logic 274 supplies control signals to logic analyzer 260 and assists with retrieving data and status from the logic analyzer.

Detailed Description Text (56):
Typically, a JTAG port is used either to program a PLD or to assist with testing a circuit board on which PLDs are located. Advantageously, it is realized that a JTAG port has traditionally been unused during the design and debugging of a particular PLD. Thus, it is further realized that a JTAG port on a PLD is under utilized and may be used during debugging of a PLD as a means of communicating with and controlling an embedded logic analyzer of the present invention. Advantageously, a standard JTAG port is used to facilitate debugging of a programmable logic device that includes an embedded logic analyzer. Two particular embodiments for implementing control logic 274 to facilitate control of an embedded logic analyzer by a JTAG port are described below in FIGS. 11-13 and 14-17, respectively.

Detailed Description Text (57):
FIG. 7 illustrates the input and output signals for embedded logic analyzer 260 according to one embodiment of the present invention. Signals Dataln 280 are the signals specified by the user in step 108 that are to be monitored for the purpose of debugging the PLD. Signal SetDelay 281 is a control line that causes register 310 to be loaded by the value of signal Delay 282 which indicates the number of samples to be captured following a breakpoint. Signal Delay 282 indicates the number of samples to be captured following a breakpoint and is received from computer system 18 after being specified by the user. Signal Breakpoint 283 indicates to the logic analyzer when a breakpoint has occurred. This signal is generated from trigger comparator 306 within the logic analyzer, or may be generated within the user designed logic.

Detailed Description Text (60):
FIG. 8 illustrates embedded logic analyzer 260 according to one embodiment of the present invention. A logic analyzer to be embedded within a PLD may be implemented in a wide variety of manners depending upon the type of PLD, signal type and number to be monitored, depth of data desired, memory available, control signals from the user's computer and preferences of the designing engineer, etc. By way of example, logic analyzer 260 is one particular example of how such a logic analyzer may be implemented. The embedded logic analyzer is controlled by the user from a computer external to the PLD and operates to capture any of a variety of internal signals that the user wishes. In this embodiment of the invention, logic analyzer 260 includes control state machine 302, trigger register 304, trigger comparator 306, registers 308 and 310, counters 312-316, comparators 320, 322 and sample memory 324.

Detailed Description Text (62):
Control state machine 302 may be any suitable control structure for controlling the embedded logic analyzer and is described in greater detail in FIG. 9. Preferably, state

machine 302 is implemented in programmable logic using any of a variety of look-up tables, embedded memory blocks, ROM registers, etc. Input signal DelayDone is received from comparator 320 and indicates that the total number of samples requested by the user have been captured. Signals NextReq, StopReq, RunReq, DoneDump, Clock and Clear have been described above. Input signal Breakpoint to state machine 302 is received from trigger comparator 306 via register 308.

Detailed Description Text (76):
As described above with reference to FIG. 6, a preferred embodiment of the invention uses JTAG port 272 along with control logic 274 and signals 275 for controlling logic analyzer 260. It is realized that use of a JTAG port for control of a logic analyzer would be advantageous in that a JTAG port is often already present on a PLD. Furthermore, use of a JTAG port would obviate the need to add extra, dedicated debugging control pins. Furthermore, many manufacturers of PLDs already have facilities for connecting and communicating through a JTAG port of a PLD. For example, Altera Corporation of San Jose, Calif. uses an internal product known as "Byte Blaster" to program a PLD through a JTAG port. For these reasons and others, it is realized that use of a JTAG port to control an embedded logic analyzer would be advantageous. Nevertheless, how to implement such control using a JTAG port is not intuitively obvious for a variety of reasons.

Detailed Description Text (77):
For background, more detail on use of a JTAG port will now be provided. A JTAG testing device tests a hardware device having a JTAG port by basically taking over control of the pad ring of the device. In other words, the JTAG tester takes over control of the drivers for each pin, effectively isolating the core of the device from the outside world. Using the JTAG port of the device then, the JTAG tester is able to put each pin into one of three states: drive, sense, or tri-state. The JTAG testing device is primarily used in an EXTEST mode to perform a full board test of the physical connections between devices on a board. By driving a pin on one device to output a signal, and sensing an input on another device on the board, the JTAG tester in this mode is able to test the physical connection between the devices while on the board. As such, EXTEST mode would be unsuitable for controlling an embedded logic analyzer. The INTEST mode is used less often and is used to internally test a device. As above, the JTAG testing device takes control of each pin driver and isolates the core. Test signals may then be driven into the core and output signals may be sampled and their accuracy determined.

Detailed Description Text (78):
Unfortunately, because the INTEST mode disconnects the core of the device from the outside world, the PLD is not being tested in a real-world environment on the circuit board. As previously explained, it is often necessary to test a PLD within the real-world environment of an operating circuit board in order to track down elusive malfunctions. Furthermore, a JTAG port is only a 10 MHz serial port, and is thus not able to provide the high-speed volume of data that might occur in a real-world environment. Thus, actual high speed operating conditions would be desirable. Additionally, during a JTAG test, the engineer provides contrived test vectors that may not be representative of real-world signals that the PLD would receive in a true operating environment. For these reasons and others, it may not be particularly desirable to attempt to control an embedded logic analyzer of a PLD using the INTEST mode of the JTAG port. Nevertheless, it is realized that using a JTAG port in some fashion to control an embedded logic device would be desirable. Advantageously, the present invention contemplates two embodiments by which JTAG port 272 controls embedded logic analyzer 260 of a PLD 16. Advantageously, a JTAG port is used to control the embedded logic analyzer while the PLD in which the logic analyzer is embedded is allowed to operate on the circuit board in a real-world environment. These two embodiments are presented below in FIGS. 11-13 and FIGS. 14-17, respectively.

Detailed Description Text (80):
FIG. 11 illustrates a first embodiment by which JTAG port 272 controls embedded logic analyzer 260 of PLD 16 using groups of unbonded I/O cells 504 and 506. Logic analyzer 260 is embedded in core 502 of PLD 16 and has a system clock 288. Cells 504 deliver signals 514 to the logic analyzer, and cells 506 receive signals 516 from the logic analyzer. Signals 275 represent signals from JTAG port 272 to and from I/O cells 504 and 506. Included are: signal TDI that connects to serial data in (SDI) of the first input cell 504; TDO that connects to serial data out (SDO) of the last output cell 506; and control signals such as Shift 680, Clock 682, Update 684, and Mode 686 that are provided to each cell as required. In this embodiment, JTAG-enabled I/O cells 504 are used to control logic analyzer 260 via input signals 514. Output data and status

information signals 516 from logic analyzer 260 is connected to JTAG-enabled I/O cells 506.

Detailed Description Text (88):
FIG. 13 illustrates an unbonded I/O cell 504 according to this first embodiment of JTAG control. Additionally included in cell 504 is gate 702 and debug RAM bit 704. In this embodiment, mode 686 places PLD 16 into its normal mode of operation so that logic analyzer 260 can capture real-world data. In this mode, pins of the PLD are not isolated from core 502. For unbonded I/O cell 504, however, it is still desirable to be able to use JTAG port 272 to provide a control signal to embedded logic analyzer 260. To these ends, gate 702 and debug RAM bit 704 are provided. Bit 704 is always set; therefore, the output of gate 702 is a logic "1" which directs multiplexer 640 to always produce its output data from update register 630. Data from register 630 had previously been loaded from capture register 620 which received its data originally from serial data in 672 (after JTAG port 272 has caused data to be shifted through the cells). In this fashion, serial data provided by JTAG port 272 is eventually output by multiplexer 640 and serves as input signal 604 to core 502. Input signal 604 may be used to provide either a control signal or a data input signal for logic analyzer 260. Advantageously, the device may be operated in normal mode and all pins that are bonded to I/O cells are still connected to core 502. Furthermore, logic analyzer 260 is allowed to be controlled via JTAG port 272 using the JTAG Sample/Preload instruction that places control information into capture registers 620-624.

Detailed Description Text (90):
FIG. 14 illustrates a second embodiment by which JTAG port 272 controls embedded logic analyzer 260 using a test data register 802. In this embodiment, a user implemented test data register 802 is used to provide control signals to, and to receive data an d status form, logic analyzer 260. This embodiment is particularly useful if no unbonded I/O cells are available. It relies upon extra user-supplied logic in test data register 802 instead of using unbonded I/O cells. In addition, this embodiment provides an extra signal Runtest(user) that allows logic analyzer 260 to know when the JTAG state machine has entered the Runtest state. Register 802 includes any number of stimulus cells 804 used to control logic analyzer 260 and any number of sense cells 805 used for retrieving data and status from the logic analyzer. Control signals 806 include signal TDI(user) which is presented to the first stimulus cell and then shifted through all of the cells. Also included are the control signals Shift(user), Clock(user), Update(user), and Runtest(user); these signals are presented globally to each cell 804 or 805. Signal TDO(user) 807 is received from the from sense cell 805 and presented to JTAG port 272 to become signal TDO.

Detailed Description Text (91):
In this embodiment, control signals TDI(user), Shift(user), Clock(user), and Update(user) are analogous to signals 672, 680, 682 and 684 from the embodiment shown in FIG. 13 except that these control signals in this second embodiment are driven into core 502 instead of being presented to I/O cells. Advantageously, using this uncommon approach of driving JTAG signals directly into the core, control of an embedded logic analyzer is achieved without using extra pins or I/O cells of the PLD (aside from the JTAG port pins). Signal TDO(user) is analogous to signal 674 of the embodiment of FIG. 13 except that signal TDO(user) originates from core 502 instead of from an I/O cell 504.

Detailed Description Text (96):
FIGS. 17A, 17B illustrate an alternative embodiment in which any number of logic analyzers embedded within a device are controlled using a JTAG port. As PLDs become larger and larger, it is possible that each megafinction within the device may contain its own embedded logic analyzer. It would be desirable to be able to control any number of embedded logic analyzers using a JTAG port using any of the embodiments discussed herein. In one particular implementation, the second embodiment discussed above in FIGS. 14-16 works well.

Detailed Description Text (97):
Digressing for a moment, it is noted that control of one of two embedded logic analyzers may be achieved using a Select signal generated from JTAG port 272. As is known in the art, private user instructions may be loaded into the JTAG port. In this embodiment, a UserA instruction and a UserB instruction may be provided. Control information destined for a first logic analyzer is loaded into the UserA instruction; control information destined for a second logic analyzer is loaded into the UserB instruction. When UserA is loaded, signal Select goes high, when UserB is loaded, Select goes low. Signal Select is then combined with and qualifies the control signals

from the JTAG port to be directed to either a first or a second test data register that control respectively, the first or the second embedded logic analyzer. As is known in the art, a single signal (for example, Select) can enable or disable a control signal for a logic analyzer using a simple combination of AND gates, inverters, etc. For example, when Select is a logic "1", control signals are directed to the first logic analyzer and outputs are received from it. The second logic analyzer is selected when Select is a logic "0". For more than two logic analyzers to be controlled, it is useful to use an embodiment such as will now be described.

Detailed Description Text (109):
Other embodiments are also possible. The first embodiment presented above in FIGS. 11-13 or the second embodiment presented above FIGS. 14-16 may be used exclusively to control a logic analyzer or they may be combined to provide control. If a sufficient number of extra unbonded I/O cells are available, it may be desirable to use the first embodiment exclusively. This is especially true if it would be difficult to insert extra logic into the device. If insufficient I/O cells are available, it may be desirable to use the second embodiment, as long as the addition of the extra logic required by test data register 802 is not a problem. Using exclusively the first embodiment, a Clock signal can be provided to the embedded logic analyzer by using input signal 604. To provide this Clock signal, alternating 1's and 0's are shifted into one particular I/O cell and then loaded one at a time to provide an alternating pulse. Each new bit, though, must be scanned in through an entire set of registers before the bit can be provided as a Clock signal. For this reason, this technique of providing a Clock signal to the embedded logic analyzer using the first embodiment is not extremely efficient.

Detailed Description Text (110):
In a more optimal solution, a combination of the first and second embodiments are used. In this solution, the extra signal Runtest(user) available in the second embodiment is used to provide a Clock signal to the embedded logic analyzer. Upon transition of this Clock signal, the logic analyzer is instructed to look at the control signals arriving from input signals 604 of the various I/O cells 504 that have been implemented using the first embodiment. The signal Runtest(user) can be made to provide clock pulses simply by causing JTAG port 272 to enter this state and then back out in an alternating fashion. Using this technique, more efficient control is provided yet extra unbonded I/O cells may still be used to provide the actual control information to the logic analyzer.

Detailed Description Text (115):
Although the foregoing invention has been described in some detail for purposes of clarity of understanding, it will be apparent that certain changes and modifications may be practiced within the scope of the appended claims. For instance, a logic analyzer may be embedded in any suitable device or circuit board that lends itself to being programmed. Also, the present invention is applicable to any type of EDA tool that is able to compile a user design. Although only one example of compilation of a logic analyzer is presented, variations on this compile technique may occur depending upon the device for which the design is being compiled and still take advantage of the present invention. Furthermore, the specific logic analyzer circuit shown is exemplary; other circuits may also be used to implement a logic analyzer. An interface to the logic analyzer from a computer may use any number of pins and any type of protocol such as serial, parallel, etc. A JTAG port may control one or more embedded logic analyzers using either the first or second control embodiments described, or a combination of the two. Therefore, the described embodiments should be taken as illustrative and not restrictive, and the invention should not be limited to the details given herein but should be defin by the following claims and their full scope of equivalents.

Other Reference Publication (1):
Robert R. Collins, "Overview of Pentium Probe Mode, " (www.x86.org/articles/probemd/ProbeMode.htm), Aug. 21, 1998, 3 pages.

CLAIMS:

4. A method for reprogramming a programmable logic device (PLD) in a system, said method comprising: receiving an electronic design intended for said PLD; specifying internal signals of said electronic design to monitor; specifying a first breakpoint; compiling said electronic design with said internal signals, said first breakpoint and a logic analyzer to produce a first design file; programming said PLD with said first design file, said logic analyzer being embedded in said PLD and arranged to store said internal signals before occurrence of said first breakpoint; specifying a second

breakpoint; recompiling said electronic design with said internal signals, said second breakpoint and said logic analyzer to produce a second design file; and reprogramming said PLD with said second design file, said logic analyzer being embedded in said PLD and arranged to store said internal signals before occurrence of said second breakpoint, whereby said PLD is reprogrammed while in said system.

5. A method as recited in claim 4 further comprising: specifying a second set of internal signals of said electronic design to monitor; recompiling said electronic design with said second set of internal signals, said first breakpoint and said logic analyzer to produce a third design file; reprogramming said PLD with said third design file, said logic analyzer being embedded in said PLD and arranged to store said second set of internal signals before occurrence of said first breakpoint.

6. A method for receiving sample data from a logic analyzer embedded within a programmable logic device (PLD), said method comprising: establishing communication with a logic analyzer embedded within a programmable logic device (PLD); specifying a breakpoint indicative of the state of at least one signal within said PLD; indicating to said logic analyzer to continuously store internal signals of said PLD in a memory of said logic analyzer such that said internal signals are stored before the occurrence of said breakpoint; and receiving said stored internal signals from said logic analyzer, said stored signals representing at least signals stored before said breakpoint, whereby said stored internal signals may be viewed on a user computer.

WEST

End of Result Set

□ Generate Collection Print

L6: Entry 8 of 8 File: USPT Apr 7, 1998

DOCUMENT-IDENTIFIER: US 5737520 A
TITLE: Method and apparatus for correlating logic analyzer state capture data with associated application data structures

Brief Summary Text (20):
The methods of the present invention are operable on a general purpose computer. Such a computer may be either embedded within a logic analyzer or external to a logic analyzer operating in a post processing mode to analyze and display buffer and protocol information in the captured states retrieved from a logic analyzer. In the alternative, portions of the present invention may be embedded within custom hardware assist electronic circuits to enhance the performance of the analysis and recognition methods of the present invention. Such custom assist circuits may be integrated within the user interface control circuits and software of a logic analyzer.

Drawing Description Text (3):
FIG. 2 is a block diagram depicting a self-contained logic analyzer including an embedded processor and display on which the methods of the present invention are operable;

Detailed Description Text (3):
FIG. 1 is a block diagram of a typical computing system 100 to which a logic analyzer 1 is connected via probes 2 to aid in locating the root cause of a data corruption problem. System 100 includes CPU 102, connected to main memory 106 via host/PCI/cache bridge 104. CPU 102 fetches programmed instructions from main memory 106, and manipulates data in main memory 106, through host/PCI/bridge 104. Peripheral I/O devices are accessible by CPU 102 through host/PCI/cache bridge 104 on PCI bus 150. SCSI adapter 108, LAN adapter 110, bridge 112, and graphics adapter 114 are typical of devices attached to CPU 102 through the PCI bus 150. SCSI adapter 108 permits connection of the CPU 102 through PCI bus 150 to a plurality of storage devices 116 (such as disk, tape, and CDROM) via SCSI bus 152. LAN adapter 110 adapts signals on LAN 154 for use with CPU 102 via PCI bus 150. Bridge 112 extends the high speed PCI bus 150 to other lower cost, lower performance busses for attachment of additional (typically older) I/O peripheral devices. Graphics adapter 114 connects CPU 102 through PCI bus 150 to high performance graphic display screens.

Detailed Description Text (7):
Logic analyzer 1 is connected to PCI bus 150 via probes 2 to monitor and capture state logic information exchanged between CPU 102, main memory 106, and the intelligent I/O adapters 108-114. The methods of the present invention are operable on logic analyzer 1 to aid the engineer in analyzing the exchange of information over PCI bus 150 pertaining to buffers and buffer descriptors. As discussed above, such analysis at the exchange of information between CPU 102 and one or more intelligent I/O adapters 108-114, as captured in state logic data by logic analyzer 1, may be useful in isolating the root cause of a data corruption failure in such a computing system.

Detailed Description Text (9):
The methods of the present invention (discussed in detail below) are operable on a computer in conjunction with a logic analyzer. Modern logic analyzers incorporate many (if not all) aspects of a general purpose computer as shown in FIG. 2. The methods of the present invention are therefore operable directly within a logic analyzer having such general purpose computational and display capabilities. Logic analyzer 1 as shown in FIG. 2 receives state logic data from probes 2 which is processed and stored as required by state acquisition element 208 in state memory element 210. Processor 200 controls state acquisition element 208 to configure the state acquisition parameters

and to retrieve the resultant state logic data from state memory 210. As in most general purpose computing systems, processor 200 is connected to user input I/O 202 (such as keyboards and pointer devices) to receive user inputs and parameters. Processor 200 is also connected to display 204 to present information to the user of the logic analyzer. RAM/ROM memory 206 is used by processor 200 to manipulate variables and to store control program instructions. The methods of the present invention may therefore be integrated with the standard control programs in a logic analyzer, they may be stored and retrieved from RAM/ROM memory 206, execute on processor 200, display their results on display 204, and receive user input from user input I/O 202.

Detailed Description Text (10):
In the alternative, the methods of the present invention may be operated on a general purpose computer system 300 of FIG. 3 connected to logic analyzer 1 via bus 350 simply to retrieve the captured state logic data therefrom. As shown in FIG. 3, logic analyzer 1 receives state logic data from probes 2 under the control of state acquisition logic 312. The captured states are stored in state memory 314. Control element 316 (typically a computer) controls the operation of logic analyzer 1 and can communicate to an attached host computer system 300 through host interface 318 over bus 350. Bus 350 may be, for example, a SCSI bus, a LAN, an IEEE 488 bus, etc. as appropriate to the particular logic analyzer 1 and computer system 300. Computer system 300 includes CPU 302, connected to RAM/ROM memory 306, user input I/O 308 and display 310. Peripheral I/O element 304 connects CPU 302 to bus 350 for exchange of data with logic analyzer 1. In particular, computer system 300 retrieves captured state logic data from stand alone logic analyzer 1 via bus 350 then post-processes the retrieved data under the methods of the present invention to aid the engineer in locating the root cause of a data corruption error in the system under test (not shown). The methods of the present invention are therefore operable in a stand alone general purpose computer system connected to a stand alone logic analyzer. One of ordinary skill will also recognize that computer system 300 may retrieve captured state logic data via bus 350 or any other similar transfer mechanism. For example, the captured data may be written to a floppy disk on logic analyzer 1 and physically transferred to computer system 300.

Detailed Description Text (11):
One of ordinary skill in the art will recognize that the methods of the present invention may be applied within general purpose logic analyzers as well as embedded within special purpose protocol or bus analyzer devices or software modules.

WEST

□ Generate Collection Print

L7: Entry 2 of 4 File: USPT Oct 1, 2002

DOCUMENT-IDENTIFIER: US 6460148 B2
TITLE: Enhanced embedded logic analyzer

Abstract Text (1):
Embedding a logic analyzer in a programmable logic device allows signals to be captured both before and after a trigger condition (breakpoint). A logic analyzer embedded within a PLD captures and stores logic signals. It unloads these signals for viewing on a computer. Using an electronic design automation (EDA) software tool running on a computer system, an engineer specifies signals of the PLD to be monitored, a breakpoint, total number of samples to be stored, number of samples to be captured after the breakpoint occurs, and a system clock signal. The EDA tool automatically inserts the logic analyzer into the electronic design of the PLD which is compiled and downloaded to configure the PLD. Using an interface connected between the PLD and the computer, the EDA tool commands the embedded logic analyzer to run. Signals are stored continuously while running in a ring buffer RAM memory. Once the breakpoint occurs, more samples are captured if desired, in addition to those signals captured before breakpoint. The EDA tool directs the logic analyzer to unload the data from its capture buffer for display on a computer. The breakpoint and sample number can be changed without recompiling. A JTAG port controls the logic analyzer. Inputs and outputs of the logic analyzer are routed to unbonded JTAG-enabled I/O cells. Alternatively, a user-implemented test data register provides a JTAG-like chain of logic elements through which control and output information is shifted. Stimulus cells provide control information to the logic analyzer, and sense cells retrieve data from the logic analyzer.

Parent Case Text (1):
This application claims priority of U.S. provisional patent application No. 60/065,602, filed Nov. 18, 1997, entitled "Enhanced Embedded Logic Analyzer" which is incorporated by reference. This application is a continuation of U.S. patent application Ser. No. 09/186,607 filed Nov. 16, 1998 now U.S. Pat. No. 6,286,114, which in turn is a continuation-in-part of U.S. patent application Ser. No. 08/958,435, filed Oct. 27, 1997 now U.S. Pat. No. 6,182,247, entitled "Embedded Logic Analyzer For A Programmable Logic Device" which is incorporated by reference.

Parent Case Text (2):
This application is related to U.S. Pat. No. 6,247,147, entitled "Enhanced Embedded Logic Analyzer," which is hereby incorporated by reference.

Brief Summary Text (2):
The present invention relates generally to analysis of a hardware device in connection with a computer system. More specifically, the present invention relates to a logic analyzer that is automatically embedded within a hardware device for purposes of debugging.

Brief Summary Text (8):
One approach to debugging a hardware device within a working system is to use a separate piece of hardware equipment called a logic analyzer to analyze signals present on the pins of a hardware device. (For example, the HP1670A Series Logic Analyzer from Hewlett-Packard Company.) Typically, a number of probe wires are connected manually from the logic analyzer to pins of interest on the hardware device in order to monitor signals on those pins. The logic analyzer captures and stores these signals. However, the use of an external logic analyzer to monitor pins of a hardware device has certain limitations when it comes to debugging such a device. For example, such an external logic analyzer can only connect to and monitor the external pins of the hardware device. Thus, there is no way to connect to and monitor signals that are internal to the hardware device. Unfortunately, when programming a hardware device such as a PLD,

it would be useful to be able to monitor some of these internal signals in order to debug the PLD.

Brief Summary Text (11):
In some cases, it is possible for an engineer to employ a conventional logic analyzer to study an internal signal of a PLD. This may be accomplished by, for example, an engineer modifying his design so that a normally internal signal is routed temporarily to an output pin of the PLD. The design is then recompiled. The engineer then attaches a probe to this output pin in order to monitor the "internal" signal. Unfortunately, the engineer must recompile his design and reprogram the PLD in order to view this internal signal. Also, when debugging is complete, the engineer must again rewrite the design to remove the internal signal from the output pin, recompile the design and fully reprogram the PLD again. This can be a tedious process.

Brief Summary Text (12):
Even if an engineer is successful in routing an internal signal to an output pin of a PLD, with certain integrated circuit packages it may be extremely difficult to attach an external logic analyzer. For an integrated circuit in a dual in-line package it may be relatively straightforward to attach the probes of a logic analyzer to the top of the package as long as the package is in an easily accessible location on a circuit board. However, if the package is in a difficult to reach location because of device crowding, it may be difficult to physically attach logic analyzer probes to particular output pins of interest. Even more troublesome are integrated circuits with rows of miniature contacts located on the top of the package (e.g., "flip chips"). It is difficult to attach logic analyzer probes to particular outputs of interest with this type of package. Some integrated circuit are encased in a ball grid array package with the contacts located on the bottom of the package up against the circuit board; for these packages, it may be nearly impossible to attach logic analyzer probes to these small contacts located on the underside of the package. Thus, use of an external logic analyzer has shortcomings even if an internal signal can be routed to a pin of a device.

Brief Summary Text (13):
U.S. patent application Ser. No. 08/958,435 entitled "Embedded Logic Analyzer For A Programmable Logic Device" discloses an advantageous apparatus and techniques that allow an embedded logic analyzer to flexibly analyze internal signals of interest in an electronic design, such as within a programmable logic device (PLD). Nevertheless, there is room for improvement in the analysis of internal signals of a PLD for debugging purposes.

Brief Summary Text (16):
Various prior art efforts present partial solutions, but each have their drawbacks. For example, external logic analyzers available from the Hewlett-Packard Company allow capture of signal data before a trigger condition (or breakpoint) occurs. Unfortunately, these external logic analyzers suffer from many of the disadvantages associated with external logic analyzers discussed above. Actel Corporation of Sunnyvale, Calif. provides two probes within a programmable logic device that are able to monitor two different signals, but these signals must be prespecified by the user and may not be flexibly reassigned to other signals. In addition, the Actel probes provide constant monitoring of particular signals, but do not allow capture of relevant signal data in relation to a specified breakpoint.

Brief Summary Text (17):
Therefore it would be desirable to have an apparatus and technique that would allow a logic analyzer embedded within a programmable logic device to flexibly capture internal signals both before and after a specified breakpoint.

Brief Summary Text (18):
Furthermore, it would be desirable to have an apparatus and technique that would efficiently and flexibly control a logic analyzer embedded within a programmable logic device. As explained below, although various options are available for controlling such an embedded logic analyzer, none of the prior art techniques are optimal. By way of background, a brief explanation of the design and manufacturing phases of a PLD and a circuit board will be provided first.

Brief Summary Text (19):
As described earlier in this section, a design engineer designs a PLD and programs such a device using an electronic design automation tool. In the course of this design phase, the design engineer may perform numerous design-program-debug iterations before

the design is complete and the PLD ready for mass manufacturing. The design engineer often uses a simulation and/or a timing analysis to assist in debugging the electronic design of the PLD. It is also conceivable that a design engineer would use an embedded logic analyzer (such as disclosed in U.S. patent application Ser. No. 08/958,435) to troubleshoot the design. Once the design of the PLD is complete to the design engineer's satisfaction, the design is handed off to a product engineer for the manufacturing phase.

Brief Summary Text (21):
A full board test may then be performed to test the traces, solder connections, and other physical interfaces between components on the board. It should be pointed out that a board test may also be performed before any devices on the board are programmed or configured. It is common to use a JTAG port of a PLD or other device to test the traces and solder connections of a board during this board test. Once physical connections are tested, a complete functional test of the board is then formed to test the overall functionality of the board (i.e., to ensure that particular inputs produce the outputs expected). At this point, if a failure is detected it may be necessary to debug a particular PLD while on the board. For failures more difficult to track down, it may even be necessary to remove a PLD from the board to be debugged. In these circumstances, as previously explained, it is desirable to have an embedded logic analyzer within the PLD to facilitate debugging. During any debugging of the PLD using an embedded logic analyzer, it is necessary in some fashion to control the embedded logic analyzer, i.e., to provide it with commands and data and to receive captured data and status from it. Although various options are available, none are currently optimal.

Brief Summary Text (22):
For example, it may be possible to use existing input/output pins of a device to provide a control interface. Unfortunately, a particular design may not have enough extra input/output pins available through which an interface can be provided to control an embedded logic analyzer. It can be undesirable to require that a customer purchasing a PLD not use a certain number of input/output pins simply because the PLD may not have been designed correctly and might have to be debugged at some point.

Brief Summary Text (24):
Therefore, an apparatus and technique are further desirable that would provide simple, efficient and flexible control of an embedded logic analyzer. It would further be desirable for such a control apparatus and technique to allow testing of a PLD on a circuit board in a real-world environment.

Brief Summary Text (26):
To achieve the foregoing, and in accordance with the purpose of the present invention, a technique for embedding a logic analyzer in a programmable logic device is disclosed that allows capture of specified signal data both before and after a specified breakpoint. Also disclosed are techniques for controlling an embedded logic analyzer using a JTAG port.

Brief Summary Text (27):
The present invention provides both an apparatus and a technique by which a logic analyzer circuit is automatically embedded within a PLD, by which it captures and stores logic signals both before and after a breakpoint, and by which it unloads these signals through an interface to a computer. In a preferred embodiment, analysis of the signals is performed on the computer, with the "on-chip" logic analyzer circuit serving only to acquire the signals. The invention works especially well with a PLD because, by its very nature, a PLD is able to be programmed with a design, the design may be changed, and the PLD programmed again and again. Thus, the logic analyzer circuit may be embedded in test designs or iterations in the process of designing a final PLD. Upon successful debugging of the PLD design, the PLD chip can be reprogrammed without the logic analyzer circuit, or the circuit can be left on the chip.

Brief Summary Text (28):
In one embodiment of the invention, using an electronic design automation (EDA) software tool running on a computer system, an engineer specifies signals of the PLD to be monitored, specifies the number of samples to be captured, specifies a system clock signal, and specifies not only a breakpoint, but also the number of samples needed prior to the breakpoint. (Alternatively, total samples could be specified and/or samples needed after a breakpoint.) The EDA tool then automatically inserts the logic analyzer circuit into the electronic design of the PLD which is compiled and downloaded to configure the PLD. Using an interface connected between the PLD and the computer,

the EDA tool communicates with the embedded logic analyzer in order to instruct the logic analyzer to run and to begin capturing data. Once a breakpoint occurs, the logic analyzer determines if more samples need to be captured after the breakpoint. The EDA tool then directs the logic analyzer to unload the data from sample memory and then displays the data on the computer. The logic analyzer circuit may then run again to capture another sequence of sample values.

Brief Summary Text (30):
Often, a JTAG port is used either to program a PLD or to assist with testing a circuit board on which PLDs are located. Advantageously, it is realized that a JTAG port has traditionally been unused during the design and debugging of a particular PLD. Thus, it is further realized that a JTAG port on a PLD is under utilized and may be used during debugging of a PLD as a means of communicating with and controlling an embedded logic analyzer of the present invention. Advantageously, the standard JTAG port is used to facilitate debugging of a programmable logic device that includes an embedded logic analyzer. Use of a JTAG port avoids adding dedicated debugging control pins. In a first embodiment for controlling an embedded logic analyzer using a JTAG port, inputs and outputs of the logic analyzer are routed to unbonded JTAG-enabled I/O cells. Cells that will provide control signals are tricked into thinking they are in INTEST mode so that signals may be input, yet the rest of the device operates as in a real-world environment. In a second embodiment, a user-implemented test data register provides a JTAG-like chain of logic elements through which control and output information is shifted. Stimulus cells provide control information to the logic analyzer, and sense cells retrieve data from the logic analyzer.

Brief Summary Text (31):
The present invention provides many advantages over the prior art. Use of an embedded logic analyzer in a PLD allows debugging of the device in the system in which it is operating and under the actual conditions that might produce a malfunction of the PLD. The technique of the present invention automatically embeds a logic analyzer circuit into a PLD so that an engineer may debug any logic function within the device. The embedded logic analyzer is able to capture any internal signals specified by the engineer; the breakpoint can also include any specified internal signals. Through the use of memory within the embedded logic analyzer and an interface to the computer, any number and depth of signals can be monitored within the device and then transmitted to the computer at a later time for analysis. In one embodiment of the invention, a JTAG port is used to program the embedded logic analyzer and to transmit captured signal information to the computer.

Brief Summary Text (32):
Advantageously, while debugging a PLD design in a system, an engineer may use the EDA tool to specify new signals to monitor and/or new breakpoints. The engineer can then reprogram the device while it is within its intended system with a modified logic analyzer circuit very rapidly in order to debug a different portion of the device or to change the triggering conditions. This ability to reprogram an embedded logic analyzer on the fly has many advantages over built-in debugging hardware on custom chips that may not be dynamically reprogrammed. This ability to reprogram also has advantages over external logic analyzers that can only monitor the external pins of a hardware device. Furthermore, once an engineer has finished debugging the device with the embedded logic analyzer, the EDA tool may be used to generate a final configuration output file without the logic analyzer that represents the engineer's final working design. Thus, the logic analyzer need not be part of the final design and take up space on the PLD.

Brief Summary Text (33):
The present invention is applicable to a wide range of hardware devices, and especially to PLDs. A PLD in particular may be implemented using a wide variety of technologies, including SRAM technology and EEPROM technology. PLDs based upon SRAM technology are especially advantageous in that they may have additional embedded memory that can be used by the embedded logic analyzer to capture a large number of, and a greater depth of signals. Furthermore, an embedded logic analyzer that is designed and inserted automatically by an EDA tool means that an engineer does not require an external logic analyzer as a separate piece of equipment. Furthermore, the engineer may use the computer on which he or she is creating a design for the PLD to also control and configure the embedded logic analyzer and to review its results.

Brief Summary Text (34):
In one embodiment of the present invention, a number of pins on the PLD are dedicated interface pins for communication with the user computer. Because these pins are dedicated for the interface, and are known ahead of time, they may be routed to an

easily accessible location or port on a circuit board, such that a debugging interface cable may be connected from the user computer to these pins extremely easily. This technique is especially advantageous where pins or contacts of a particular integrated circuit in a package may be difficult or nearly impossible to reach. Because the embedded logic analyzer of the present invention may be configured to monitor any internal or external signals of the PLD, all of these monitored signals are available for analysis through these interface pins. In other words, it is not necessary to physically connect a probe to a particular external pin of interest because any signal within the PLD can be monitored, stored within the memory of the embedded logic analyzer and then later uploaded to the user computer for analysis through these dedicated interface pins.

Brief Summary Text (35):
Additionally, an embedded logic analyzer can be used with PLDs that are configured to near capacity. An engineer can temporarily remove a portion of the design unrelated to the problem under analysis, embed a logic analyzer circuit, and then debug the PLD. Once the PLD has been debugged, the engineer may then remove the embedded logic analyzer and reinsert that section of the design that he had temporarily removed.

Drawing Description Text (5):
FIGS. 3A and 3B are a flowchart describing one technique by which a logic analyzer is programmed, embedded within a device, captures data and dumps data for viewing by a user.

Drawing Description Text (7):
FIG. 5 is another view of the block diagram of FIG. 1, showing a programmable logic device having an embedded logic analyzer within an electronic system.

Drawing Description Text (8):
FIG. 6 is a more detailed block diagram of a programmable logic device having an embedded logic analyzer.

Drawing Description Text (9):
FIG. 7 illustrates an embedded logic analyzer showing its inputs and outputs according to one embodiment of the present invention.

Drawing Description Text (10):
FIG. 8 is a block diagram of an embedded logic analyzer circuit according to one embodiment of the present invention.

Drawing Description Text (11):
FIG. 9 is a symbolic view of the operation of the control state machine of the embedded logic analyzer.

Drawing Description Text (13):
FIG. 11 illustrates a first embodiment by which a JTAG port controls an embedded logic analyzer using unbonded I/O cells.

Drawing Description Text (16):
FIG. 14 illustrates a second embodiment by which a JTAG port controls an embedded logic analyzer using a test data register.

Drawing Description Text (19):
FIGS. 17A and 17B illustrate an alternative embodiment in which any number of logic analyzers embedded within a device are controlled using a JTAG port.

Detailed Description Text (22):
Within the context of the present invention, any of the above compile techniques may be modified in order to produce an embedded logic analyzer. As will be discussed in greater detail below with reference to FIG. 4, the compilation of a PLD is modified in order to insert a logic analyzer into a user's design.

Detailed Description Text (25):
Embedded Logic Analyzer

Detailed Description Text (29):
These signals to be monitored may be specified in a wide variety of ways. By way of example, a hierarchical path name for each signal may be specified, or a graphical user interface may be used to view a particular design file and to select a signal or point

from within that file to be monitored. In one alternative embodiment, the user may also specify which pins of the device will be used as an interface to the user computer, i.e., those pins to be used to send control information to the embedded logic analyzer within the PLD and to upload captured information from the logic analyzer to the user computer. Preferably, though, the pins to be used as an interface are already known, such as a JTAG port of a device.

Detailed Description Text (33):
In step 114, a breakpoint is specified. A breakpoint may include any number of signals to monitor and the logic levels that those signals must have to trigger the logic analyzer, i.e., the breakpoint describes a particular state of the device. A breakpoint may be as simple as one signal changing state, or may be a complex pattern of signals or a sequence of patterns that occur before to trigger the logic analyzer. Also, a breakpoint need not be specified in all cases; if not, the logic analyzer immediately begins capturing data upon running. Advantageously, the breakpoint can be changed at any time by the user through the use of the EDA tool, and a new breakpoint can be downloaded to the embedded logic analyzer in the device without having to recompile all of the device design files. By allowing breakpoints to be changed rapidly for a device within a system, debugging is much more efficient. Advantageously, the present invention can acquire data not only after the breakpoint, but also before it occurs.

Detailed Description Text (34):
In step 115 the user specifies the number of data samples to be captured prior to the breakpoint. Advantageously, the user may specify any number of samples to be captured prior to the breakpoint occurring, thus allowing later analysis of these prior signals to help determine the cause of a failure, error or other condition. As explained below in FIGS. 8-10, implementation of the embedded logic analyzer allows samples to be stored continuously which provides a user with any number of samples needed prior to the breakpoint. Alternatively, a user may specify the number of samples needed after the breakpoint. Because the total number of samples to be captured has been specified in step 110, it is straightforward to calculate prior samples needed based upon later samples needed, and vice-versa. The user may also specify samples needed prior to the breakpoint and samples needed after the breakpoint; total samples to be captured can then be calculated automatically.

Detailed Description Text (36):
Once the user has specified how he or she wishes the embedded logic analyzer to function, the complete design is compiled. In step 116, the user issues a compile command to compile the user's device design along with the logic analyzer design that has been specified. In a preferred embodiment of the invention, the design files are not modified during this process. The logic analyzer design is incorporated into the output files produced. In one specific embodiment, the process shown in FIG. 4 may be used to implement step 116. Of course, other similar techniques may also be used.

Detailed Description Text (37):
The result of this step is a new output file that includes the user's design with an embedded logic analyzer. A technique by which an EDA tool may insert a custom logic analyzer in a design will be discussed in greater detail below with reference to FIG. 4. Once the new output file has be en generated, in step 118 the device is reprogrammed within its system using the new output file.

Detailed Description Text (39):
The cable may attach directly to these pins, or, the signals from these pins may be routed to an easily accessible location or port on the board to which the debugging cable may easily attach. The cable will be used to transmit instructions from the computer to the embedded logic analyzer, and also to upload captured information from the logic analyzer to the computer. As discussed below, FIG. 5 shows a PLD cont aining both a user design and an embedded logic analyzer within an electronic system. A cable 28 is shown connecting the elec tronic system to an external computer.

Detailed Description Text (40):
In step 122 the user through the EDA tool requests the embedded logic analyzer to begin running with an appropriate command. Once the logic analyzer begins to run, in step 124 it begins to continuously capture data from the signals that have been specified to be monitored. Preferably, the user then manipulates the system to duplicate previous malfunctions that the user wishes to analyze The captured data is stored within memory of the PLD, and is preferably stored within dedicated memory with in the embedded logic analyzer itself. Step 126 determines weth point has occurred. In other words, the logic analyzer determines whether the state of the signals specified to be monitored are

equivalent to the e breakpoint that the user has specified. If not, then the logic analyzer continues to capture data. If so, step 128 determines whether more samples should be captured and stored after the breakpoint. Step 128 may be implemented by comparing the total number of samples desired with the number of samples that have already been stored prior to the breakpoint. If more samples are to be stored, then in step 130 the logic analyzer continues to store the desired number of samples after the breakpoint.

Detailed Description Text (46):
The actual gate level representation of a particular logic analyzer circuit will depend upon the particular device in which the logic analyzer will be embedded. By way of example, the hardware device in which to embed the logic analyzer may include any of the PLD devices available from Altera Corporation. In particular, any of the FLEX 10K, FLEX 8000, MAX 9000, or MAX 7000 devices work well. Each of these particular devices may have different features that would affect how a gate level representation for a logic analyzer is produced. For example, for a FLEX 10K device with relatively large embedded memory sections, the is embedded memory is particularly well suited for implementing a large FIFO (first in first out) memory for the logic analyzer. For a device such as the FLEX 8000 without embedded memory, the memory elements (such as SRAM flip-flops) of logic cells may be used for the memory of the logic analyzer but the FIFO buffer may have to be divided over multiple cells if the memory in a single cell is not sufficiently large to accommodate the e buffer. Similarly, a device based upon EEPROM technology may also use one or more of its logic cells for the logic analyzer's buffer. A device having large embedded memory works particularly well with the present invention because of the larger capacity for signal storage. Thus, step 206 produces a representation for a logic analyzer circuit that is to be connected to the user's design.

Detailed Description Text (49):
Of course, another embodiment of a logic analyzer circuit may use different signals and/or a greater or fewer number of interface signals. In a preferred embodiment of the invention, these interface signals to and from the logic analyzer are connected to dedicated pins on the PLD reserved for this purpose. Thus, a user will know to which pins the debugging cable should be attached. As noted, these pins not only control the embedded logic analyzer, but also receive data from it. In other embodiments, these dedicated pins may be routed to another part of the circuit board for easy attachment of a cable. In this fashion, the logic for the logic analyzer circuit created in step 206 is connected to both the user design and to interface pins of the PLD for communication with the user computer.

Detailed Description Text (50):
In step 210 the complete design created in step 208 is placed and rout ed in a fashion that will be appreciated by those of skill in the art. The output of the place and rout e step is then input to step 212 in which the output file is assembled. This output file may then be downloaded to a PLD in order to program it. Once a PLD has been programmed with this file, a user may begin use of the embedded logic analyzer in order to debug the device.

Detailed Description Text (51):
FIG. 5 is another view of programmable logic development system 10 of FIG. 1, showing a programmable logic device having an embedded logic analyzer within an electronic system. System 10 shows an electronic system 252 connected to computer system A 18 via cable 28 or other connective appliance. Electronic system 252 includes PLD 16, a component of the electronic system. PLD 16 potentially shares one or more electronic connect ions 254 with the other components and elements the at make up the electronic system. PLD 16 has been configured with a user logic design 256 and an embedded logic analyzer 260. User logic 256 is configured with a design according to the methodology described in FIG. 2, or any other suitable design methodology. Embedded logic analyzer 260 has been incorporated into PLD 16 according to one embodiment of the invention described in FIGS. 3A and 3B.

Detailed Description Text (52):
Logical connections 262 allow signals from user logic 256 to be transmitted to logic analyzer 260. These signals may include a system clock, trigger signals, signals to monitor, etc. Pins of PLD 16 are used to connect interface signals 264 from the logic analyzer to corresponding connections 266 in electronic system 252. Cable 28 is used to connect these interface signals to computer 18. Alternatively, computer 18 may be directly connected to PLD 16 to transmit interface signals 264 to the PLD. In this manner, computer 18 transmits commands and other information to embedded logic analyzer

260, and receives information from the logic analyzer without directly interrupting or affecting the functional operation of electronic system 252. PLD 16 is thus configured to perform both the functions of user logic 256 and embedded logic analyzer 260.

Detailed Description Text (53):
Those of skill in the art will appreciate that the actual interface used between logic analyzer 260 and an external computer system may take a variety of forms. The embedded logic analyzer as herein described may be controlled by an outside computer using any suitable interface on the PLD. Furthermore, the exact number of pins on PLD 16 used to control logic analyzer 260 and to receive data from it may vary depending upon the device being programmed, the overall board design, etc. It will further be appreciated that pins used may be flexibly assigned, or that dedicated interface pins may be used. For example, interface signals 264 that provide an interface between logic analyzer 260 and external pins of PLD 16 may be implemented as described in the above-referenced U.S. patent application Ser. No. 08/958,435. Other techniques for controlling an embedded logic analyzer and for receiving output data may also be used.

Detailed Description Text (54):
FIG. 6 illustrates another view of PLD 16 showing a preferred embodiment for controlling a logic analyzer using the JTAG port of the device in which the logic analyzer is embedded. Not shown for clarity within PLD 16 is user logic 256. In this preferred embodiment, interface signals 264 are implemented using a JTAG port 272 in conjunction with control logic 274 and signals 275. A JTAG (Joint Test Action Group) port 272 is implemented under the IEEE 1149.1 standard and is known to those of skill in the art. Control logic 274 provide buffering between logic analyzer 260 and JTAG port 272 for particular signals that are described below in FIG. 7. More specifically, control logic 274 supplies control signals to logic analyzer 260 and assists with retrieving data and status from the logic analyzer.

Detailed Description Text (56):
Typically, a JTAG port is used either to program a PLD or to assist with testing a circuit board on which PLDs are located. Advantageously, it is realized that a JTAG port has traditionally been unused during the design and debugging of a particular PLD. Thus, it is further realized that a JTAG port on a PLD is under utilized and may be used during debugging of a PLD as a means of communicating with and controlling an embedded logic analyzer of the present invention. Advantageously, a standard JTAG port is used to facilitate debugging of a programmable logic device that includes an embedded logic analyzer. Two particular embodiments for implementing control logic 274 to facilitate control of an embedded logic analyzer by a JTAG port are described below in FIGS. 11-13 and 14-17, respectively.

Detailed Description Text (57):
FIG. 7 illustrates the input and output signals for embedded logic analyzer 260 according to one embodiment of the present invention. Signals Dataln 280 are the signals specified by the user in step 108 that are to be monitored for the purpose of debugging the PLD. Signal SetDelay 281 is a control line that causes register 310 to be loaded by the value of signal Delay 282 which indicates the number of samples to be captured following a breakpoint. Signal Delay 282 indicates the number of samples to be captured following a breakpoint and is received from computer system 18 after being specified by the user. Signal Breakpoint 283 indicates to the logic analyzer when a breakpoint has occurred. This signal is generated from trigger comparator 306 within the logic analyzer, or may be generated within the user designed logic.

Detailed Description Text (60):
FIG. 8 illustrates embedded logic analyzer 260 according to one embodiment of the present invention. A logic analyzer to be embedded within a PLD may be implemented in a wide variety of manners depending upon the type of PLD, signal type and number to be monitored, depth of data desired, memory available, control signals from the user's computer and preferences of the designing engineer, etc. By way of example, logic analyzer 260 is one particular example of how such a logic analyzer may be implemented. The embedded logic analyzer is controlled by the user from a computer external to the PLD and operates to capture any of a variety of internal signals that the user wishes. In this embodiment of the invention, logic analyzer 260 includes control state machine 302, trigger register 304, trigger comparator 306, registers 308 and 310, counters 312-316, comparators 320, 322 and sample memory 324.

Detailed Description Text (62):
Control state machine 302 may be any suitable control structure for controlling the embedded logic analyzer and is described in greater detail in FIG. 9. Preferably, state

machine 302 is implemented in programmable logic using any of a variety of look-up tables, embedded memory blocks, ROM registers, etc. Input signal DelayDone is received from comparator 320 and indicates that the total number of samples requested by the user have been captured. Signals NextReq, StopReq, RunReq, DoneDump, Clock and Clear have been described above. Input signal Breakpoint to state machine 302 is received from trigger comparator 306 via register 308.

Detailed Description Text (76):
As described above with reference to FIG. 6, a preferred embodiment of the invention uses JTAG port 272 along with control logic 274 and signals 275 for controlling logic analyzer 260. It is realized that use of a JTAG port for control of a logic analyzer would be advantageous in that a JTAG port is often already present on a PLD. Furthermore, use of a JTAG port would obviate the need to add extra, dedicated debugging control pins. Furthermore, many manufacturers of PLDs already have facilities for connecting and communicating through a JTAG port of a PLD. For example, Altera Corporation of San Jose, Calif. uses an internal product known as "Byte Blaster" to program a PLD through a JTAG port. For these reasons and others, it is realized that use of a JTAG port to control an embedded logic analyzer would be advantageous. Nevertheless, how to implement such control using a JTAG port is not intuitively obvious for a variety of reasons.

Detailed Description Text (77):
For background, more detail on use of a JTAG port will now be provided. A JTAG testing device tests a hardware device having a JTAG port by basically taking over control of the pad ring of the device. In other words, the JTAG tester takes over control of the drivers for each pin, effectively isolating the core of the device from the outside world. Using the JTAG port of the device then, the JTAG tester is able to put each pin into one of three states: drive, sense, or tri-state. The JTAG testing device is primarily used in an EXTEST mode to perform a full board test of the physical connections between devices on a board. By driving a pin on one device to output a signal, and sensing an input on another device on the board, the JTAG tester in this mode is able to test the physical connection between the devices while on the board. As such, EXTEST mode would be unsuitable for controlling an embedded logic analyzer. The INTEST mode is used less often and is used to internally test a device. As above, the JTAG testing device takes control of each pin driver and isolates the core. Test signals may then be driven into the core and output signals may be sampled and their accuracy determined.

Detailed Description Text (78):
Unfortunately, because the INTEST mode disconnects the core of the device from the outside world, the PLD is not being tested in a real-world environment on the circuit board. As previously explained, it is often necessary to test a PLD within the real-world environment of an operating circuit board in order to track down elusive malfunctions. Furthermore, a JTAG port is only a 10 MHz serial port, and is thus not able to provide the high-speed volume of data that might occur in a real-world environment. Thus, actual high speed operating conditions would be desirable. Additionally, during a JTAG test, the engineer provides contrived test vectors that may not be representative of real-world signals that the PLD would receive in a true operating environment. For these reasons and others, it may not be particularly desirable to attempt to control an embedded logic analyzer of a PLD using the INTEST mode of the JTAG port. Nevertheless, it is realized that using a JTAG port in some fashion to control an embedded logic device would be desirable. Advantageously, the present invention contemplates two embodiments by which JTAG port 272 controls embedded logic analyzer 260 of a PLD 16. Advantageously, a JTAG port is used to control the embedded logic analyzer while the PLD in which the logic analyzer is embedded is allowed to operate on the circuit board in a real-world environment. These two embodiments are presented below in FIGS. 11-13 and FIGS. 14-17, respectively.

Detailed Description Text (80):
FIG. 11 illustrates a first embodiment by which JTAG port 272 controls embedded logic analyzer 260 of PLD 16 using groups of unbonded I/O cells 504 and 506. Logic analyzer 260 is embedded in core 502 of PLD 16 and has a system clock 288. Cells 504 deliver signals 514 to the logic analyzer, and cells 506 receive signals 516 from the logic analyzer. Signals 275 represent signals from JTAG port 272 to and from I/O cells 504 and 506. Included are: signal TDI that connects to serial data in (SDI) of the first input cell 504; TDO that connects to serial data out (SDO) of the last output cell 506; and control signals such as Shift 680, Clock 682, Update 684, and Mode 686 that are provided to each cell as required. In this embodiment, JTAG-enabled I/O cells 504 are used to control logic analyzer 260 via input signals 514. Output data and status

information signals 516 from logic analyzer 260 is connected to JTAG-enabled I/O cells 506.

Detailed Description Text (82):
FIG. 12 illustrates a prior art JTAG-enabled I/O cell 600 that provides a useful background for discussion of this embodiment. Cell 600 connects to an external PLD pin 602. Through pin 602, input signal 604 is provided to core 502 of PLD 16. Similarly, signals output 606 and output enable 608 originate within core 502 and are used to produce an output signal at pin 602. Multiplexers 610, 612 and 614 select data to be loaded into capture registers 620, 622 and 624, respectively. The capture registers are used to scan in data initially from JTAG port 272 through the I/O cells of the device. Update registers 630, 632 and 634 receive data from the capture registers and are used to perform a parallel load to the core of the device. Multiplexers 640, 642 and 644 select data from either pin 602 input, output enable 608, and output 606, respectively, or from one of the update registers to produce an appropriate signal. Multiplexer 640 produces input signal 604, multiplexer 642 produces a tri-state signal for driver 650, and multiplexer 644 provides a data signal for driver 650·which produces an output at pin 602 when enabled.

Detailed Description Text (88):
FIG. 13 illustrates an unbonded I/O cell 504 according to this first embodiment of JTAG control. Additionally included in cell 504 is gate 702 and debug RAM bit 704. In this embodiment, mode 686 places PLD 16 into its normal mode of operation so that logic analyzer 260 can capture real-world data. In this mode, pins of the PLD are not isolated from core 502. For unbonded I/O cell 504, however, it is still desirable to be able to use JTAG port 272 to provide a control signal to embedded logic analyzer 260. To these ends, gate 702 and debug RAM bit 704 are provided. Bit 704 is always set; therefore, the output of gate 702 is a logic "1" which directs multiplexer 640 to always produce its output data from update register 630. Data from register 630 had previously been loaded from capture register 620 which received its data originally from serial data in 672 (after JTAG port 272 has caused data to be shifted through the cells). In this fashion, serial data provided by JTAG port 272 is eventually output by multiplexer 640 and serves as input signal 604 to core 502. Input signal 604 may be used to provide either a control signal or a data input signal for logic analyzer 260. Advantageously, the device may be operated in normal mode and all pins that are bonded to I/O cells are still connected to core 502. Furthermore, logic analyzer 260 is allowed to be controlled via JTAG port 272 using the JTAG Sample/Preload instruction that places control information into capture registers 620-624.

Detailed Description Text (90):
FIG. 14 illustrates a second embodiment by which JTAG port 272 controls embedded logic analyzer 260 using a test data register 802. In this embodiment, a user implemented test data register 802 is used to provide control signals to, and to receive data an d status form, logic analyzer 260. This embodiment is particularly useful if no unbonded I/O cells are available. It relies upon extra user-supplied logic in test data register 802 instead of using unbonded I/O cells. In addition, this embodiment provides an extra signal Runtest(user) that allows logic analyzer 260 to know when the JTAG state machine has entered the Runtest state. Register 802 includes any number of stimulus cells 804 used to control logic analyzer 260 and any number of sense cells 805 used for retrieving data and status from the logic analyzer. Control signals 806 include signal TDI(user) which is presented to the first stimulus cell and then shifted through all of the cells. Also included are the control signals Shift(user), Clock(user), Update(user), and Runtest(user); these signals are presented globally to each cell 804 or 805. Signal TDO(user) 807 is received from the from sense cell 805 and presented to JTAG port 272 to become signal TDO.

Detailed Description Text (91):
In this embodiment, control signals TDI(user), Shift(user), Clock(user), and Update(user) are analogous to signals 672, 680, 682 and 684 from the embodiment shown in FIG. 13 except that these control signals in this second embodiment are driven into core 502 instead of being presented to I/O cells. Advantageously, using this uncommon approach of driving JTAG signals directly into the core, control of an embedded logic analyzer is achieved without using extra pins or I/O cells of the PLD (aside from the JTAG port pins). Signal TDO(user) is analogous to signal 674 of the embodiment of FIG. 13 except that signal TDO(user) originates from core 502 instead of from an I/O cell 504.

Detailed Description Text (93):
FIG. 15 illustrates a stimulus cell 804 that is an element of test data register 802.

Cell 804 includes capture register 820 and update register 822. Scan in signal 824 is received from a previous similar cell or from JTAG port 272 if this is the first stimulus cell. Scan out signal 826 is transmitted to the next stimulus cell or to the first sense cell 805 if this is the last stimulus cell. While a serial shift of information through elements of data register 802 is occurring, information arrives at cell 804 via scan in 824, is captured by register 820 and is shifted out via scan out 826. When a parallel load is performed under control of JTAG port 272, update register 822 transfers the bit stored in register 820 to logic analyzer 260. This transferred bit may then be used as a control signal for the logic analyzer.

Detailed Description Text (94):
FIG. 16 illustrates a sense cell 805 that is one element of test data register 802. Cell 805 includes multiplexer 830 and capture register 832. Scan in signal 834 is received from a previous sense cell or from the last stimulus cell 804 if this is the first sense cell. Scan out signal 836 is transmitted to the next sense cell 805 or to JTAG port 272 if this is the last sense cell. During serial scanning of information through test data register 802 signal Load(user) is a zero; scanned in bits arrive via scan in 834, are latched using register 832, and are shifted out via scan out 836. During a parallel load operation (or sense operation), signal Load(user) is a one; data and/or status arrive via multiplexer 830 and are captured by register 832. Once any number of bits are captured by cells 805 after a parallel load, the captured bits are shifted out using the serial shift mode through JTAG port 272 to computer system 18 for analysis. In this fashion, sense cells 805 are used to retrieve data and/or status from logic analyzer 260 and to present the information to a user for analysis.

Detailed Description Text (96):
FIGS. 17A, 17B illustrate an alternative embodiment in which any number of logic analyzers embedded within a device are controlled using a JTAG port. As PLDs become larger and larger, it is possible that each megafinction within the device may contain its own embedded logic analyzer. It would be desirable to be able to control any number of embedded logic analyzers using a JTAG port using any of the embodiments discussed herein. In one particular implementation, the second embodiment discussed above in FIGS. 14-16 works well.

Detailed Description Text (97):
Digressing for a moment, it is noted that control of one of two embedded logic analyzers may be achieved using a Select signal generated from JTAG port 272. As is known in the art, private user instructions may be loaded into the JTAG port. In this embodiment, a UserA instruction and a UserB instruction may be provided. Control information destined for a first logic analyzer is loaded into the UserA instruction; control information destined for a second logic analyzer is loaded into the UserB instruction. When UserA is loaded, signal Select goes high, when UserB is loaded, Select goes low. Signal Select is then combined with and qualifies the control signals from the JTAG port to be directed to either a first or a second test data register that control respectively, the first or the second embedded logic analyzer. As is known in the art, a single signal (for example, Select) can enable or disable a control signal for a logic analyzer using a simple combination of AND gates, inverters, etc. For example, when Select is a logic "1", control signals are directed to the first logic analyzer and outputs are received from it. The second logic analyzer is selected when Select is a logic "0". For more than two logic analyzers to be controlled, it is useful to use an embodiment such as will now be described.

Detailed Description Text (109):
Other embodiments are also possible. The first embodiment presented above in FIGS. 11-13 or the second embodiment presented above FIGS. 14-16 may be used exclusively to control a logic analyzer or they may be combined to provide control. If a sufficient number of extra unbonded I/O cells are available, it may be desirable to use the first embodiment exclusively. This is especially true if it would be difficult to insert extra logic into the device. If insufficient I/O cells are available, it may be desirable to use the second embodiment, as long as the addition of the extra logic required by test data register 802 is not a problem. Using exclusively the first embodiment, a Clock signal can be provided to the embedded logic analyzer by using input signal 604. To provide this Clock signal, alternating 1's and 0's are shifted into one particular I/O cell and then loaded one at a time to provide an alternating pulse. Each new bit, though, must be scanned in through an entire set of registers before the bit can be provided as a Clock signal. For this reason, this technique of providing a Clock signal to the embedded logic analyzer using the first embodiment is not extremely efficient.

Detailed Description Text (110):
In a more optimal solution, a combination of the first and second embodiments are used. In this solution, the extra signal Runtest(user) available in the second embodiment is used to provide a Clock signal to the embedded logic analyzer. Upon transition of this Clock signal, the logic analyzer is instructed to look at the control signals arriving from input signals 604 of the various I/O cells 504 that have been implemented using the first embodiment. The signal Runtest(user) can be made to provide clock pulses simply by causing JTAG port 272 to enter this state and then back out in an alternating fashion. Using this technique, more efficient control is provided yet extra unbonded I/O cells may still be used to provide the actual control information to the logic analyzer.

Detailed Description Text (115):
Although the foregoing invention has been described in some detail for purposes of clarity of understanding, it will be apparent that certain changes and modifications may be practiced within the scope of the appended claims. For instance, a logic analyzer may be embedded in any suitable device or circuit board that lends itself to being programmed. Also, the present invention is applicable to any type of EDA tool that is able to compile a user design. Although only one example of compilation of a logic analyzer is presented, variations on this compile technique may occur depending upon the device for which the design is being compiled and still take advantage of the present invention. Furthermore, the specific logic analyzer circuit shown is exemplary; other circuits may also be used to implement a logic analyzer. An interface to the logic analyzer from a computer may use any number of pins and any type of protocol such as serial, parallel, etc. A JTAG port may control one or more embedded logic analyzers using either the first or second control embodiments described, or a combination of the two. Therefore, the described embodiments should be taken as illustrative and not restrictive, and the invention should not be limited to the details given herein but should be defin by the following claims and their full scope of equivalents.

Other Reference Publication (1):
Robert R. Collins, "Overview of Pentium Probe Mode, " (www.x86.org/articles/probemd/ProbeMode.htm), Aug. 21, 1998, 3 pages.

CLAIMS:

4. A method for reprogramming a programmable logic device (PLD) in a system, said method comprising: receiving an electronic design intended for said PLD; specifying internal signals of said electronic design to monitor; specifying a first breakpoint; compiling said electronic design with said internal signals, said first breakpoint and a logic analyzer to produce a first design file; programming said PLD with said first design file, said logic analyzer being embedded in said PLD and arranged to store said internal signals before occurrence of said first breakpoint; specifying a second breakpoint; recompiling said electronic design with said internal signals, said second breakpoint and said logic analyzer to produce a second design file; and reprogramming said PLD with said second design file, said logic analyzer being embedded in said PLD and arranged to store said internal signals before occurrence of said second breakpoint, whereby said PLD is reprogrammed while in said system.

5. A method as recited in claim 4 further comprising: specifying a second set of internal signals of said electronic design to monitor; recompiling said electronic design with said second set of internal signals, said first breakpoint and said logic analyzer to produce a third design file; reprogramming said PLD with said third design file, said logic analyzer being embedded in said PLD and arranged to store said second set of internal signals before occurrence of said first breakpoint.

6. A method for receiving sample data from a logic analyzer embedded within a programmable logic device (PLD), said method comprising: establishing communication with a logic analyzer embedded within a programmable logic device (PLD); specifying a breakpoint indicative of the state of at least one signal within said PLD; indicating to said logic analyzer to continuously store internal signals of said PLD in a memory of said logic analyzer such that said internal signals are stored before the occurrence of said breakpoint; and receiving said stored internal signals from said logic analyzer, said stored signals representing at least signals stored before said breakpoint, whereby said stored internal signals may be viewed on a user computer.